Requested Patent: WO0152117A2

Title:

"DATA INTEGRATOR" SYSTEM FOR COLLECTING, FUSING AND DISPLAYING INFORMATION INCLUDING PERSISTENT CONNECTION AND STORAGE ARRANGEMENT ;

Abstracted Patent: WO0152117 ;

Publication Date: 2001-07-19 ;

Inventor(s): ZAGER MATTHEW; KING CRAIG; GRIFFITH TIMOTHY ;

Applicant(s): SCIENCE APPLIC INTERNAT CORP (US) ;

Application Number: WO2001US00793 20010110 ;

Priority Number(s): US20000480870 20000110 ;

IPC Classification: G06F17/30 ;

Equivalents: AU2780501 ;

ABSTRACT:

A computer implemented system for obtaining, storing and displaying cells of visual information such as Internet World Wide Web pages includes a display unit, a storage system for storing cells in eXtensible Markup Language (XML) format, and a communication unit for obtaining data objects or documents which can be in a plurality of different formats from the Internet and other external sources. A cell designer enables a user to create a view including a plurality of selected cells, each of which includes at least one data object. A display generator obtains the selected data objects from the storage system and generates the view for display by the display unit. A conversion unit converts data objects obtained by the communication unit into XML using eXtensible Style Language (XSL) and Document Type Definitions (DTD), and the cell designer creates the specified arrangement using XSL and DTD. The conversion unit includes a plurality of factories for converting data objects obtained by the communication unit from different respective formats into XML, and a factory manager for determining a format of a data object obtained by the communication unit and assigning it to a corresponding factory. An event manager causes the communication unit to obtain updated versions of outdated data objects from the external sources. A list unit stores a list of users and cells requested by each user, and the event manager sends notifications of the updated versions to users who have requested the specified data objects.

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
19 July 2001 (19.07.2001)

PCT

(10) International Publication Number
WO 01/52117 A2

(51) International Patent Classification[7]: G06F 17/30

(21) International Application Number: PCT/US01/00793

(22) International Filing Date: 10 January 2001 (10.01.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/480,870 10 January 2000 (10.01.2000) US

(71) Applicant: SCIENCE APPLICATIONS INTERNATIONAL CORPORATION [US/US]; 10260 Campus Point Drive, San Diego, CA 92121 (US).

(72) Inventors: KING, Craig; 9511 Red Diamond Drive, Lakeside, CA 92040 (US). GRIFFITH, Timothy; 8020 Avenida Navidad, #42, San Diego, CA 92122 (US). ZAGER, Matthew; 18557 Caminito Pasadero #386, San Diego, CA 92128 (US).

(74) Agent: GARRED, John, X.; Arter & Hadden LLP, 1100 Huntington Building, 925 Euclid Avenue, Cleveland, OH 44115 (US).

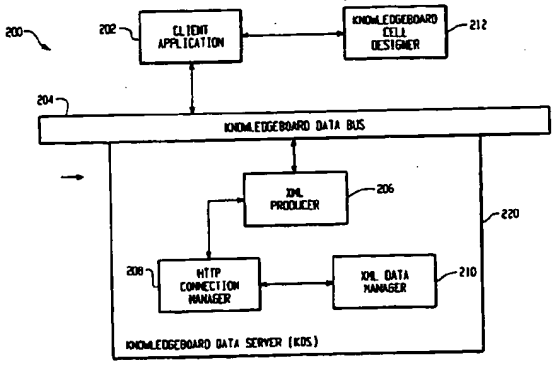
(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: "DATA INTEGRATOR" SYSTEM FOR COLLECTING, FUSING AND DISPLAYING INFORMATION INCLUDING PERSISTENT CONNECTION AND STORAGE ARRANGEMENT



(57) Abstract: A computer implemented system for obtaining, storing and displaying cells of visual information such as Internet World Wide Web pages includes a display unit, a storage system for storing cells in eXtensible Markup Language (XML) format, and a communication unit for obtaining data objects or documents which can be in a plurality of different formats from the Internet and other external sources. A cell designer enables a user to create a view including a plurality of selected cells, each of which includes at least one data object. A display generator obtains the selected data objects from the storage system and generates the view for display by the display unit. A conversion unit converts data objects obtained by the communication unit into XML using eXtensible Style Language (XSL) and Document Type Definitions (DTD), and the cell designer creates the specified arrangement using XSL and DTD. The conversion unit includes a plurality of factories for converting data objects obtained by the communication unit from different respective formats into XML, and a factory manager for determining a format of a data object obtained by the communication unit and assigning it to a corresponding factory. An event manager causes the communication unit to obtain updated versions of outdated data objects from the external sources. A list unit stores a list of users and cells requested by each user, and the event manager sends notifications of the updated versions to users who have requested the specified data objects.

WO 01/52117 A2

"DATA INTEGRATOR" SYSTEM FOR COLLECTING, FUSING AND DISPLAYING INFORMATION INCLUDING PERSISTENT CONNECTION AND STORAGE ARRANGEMENT

BACKGROUND OF THE INVENTION

Field of the Invention

The present invention generally relates to database management systems, and more specifically to a "DATA INTEGRATOR" system for collecting, fusing and displaying information including a persistent connection and storage arrangement.

Description of the Related Art

Relational databases represent a significant advancement over earlier database technologies due to the relationships established between the data tables during the storage and retrieval processes. Typically, most database systems include three basic components: structure, integrity, and manipulation.

The first of these, structure, is how information or data content is presented to the user. A database management system is considered "relational" when it is able to provide a relational view of data. This means that data which a user can access and the expressions or instructions that a user can employ to operate on that data are related. The data stored in such a database system is organized as relations, perhaps best interpreted, as tables, composed of rows (tuples) and columns (attributes). Thus, tables are a logical abstraction of what is physically stored.

Integrity, on the other hand, dictates that for every relation (i.e., table) there is a unique, primary key to identify table entries or rows. The integrity of the data is, of course, crucial. If accuracy and consistency of the data cannot be achieved, then the data may not be relied upon for decision-making purposes.

Data manipulation, the last component, may be thought of as cutting-and-pasting of tables. Data manipulation is, of course, the reason why databases exist in the first place. The superiority of manipulating data content relationally (i.e., as a whole, or in sets) is substantial. Users can logically manipulate and combine data from different segments of a database without having to specify any internal details or the order in which tables are accessed. This provides the user with a conceptual view of the database that is removed from the hardware level. Non-relational database systems, in contrast, require complex programming skills that form an inherently unreliable means to interact with databases.

Today, database management systems are everywhere - in corporate, government and academic settings, and in other shared environments. A typical installation employs one or more databases running on a server-based network system with the capability of accessing a variety of Internet based data content providers. By querying a database from a remote terminal or system, users are able to handle many of their own database-oriented application needs directly. As a result, databases are not only just another way to store and access information, but it also offers the user a number of opportunities of presenting and displaying the data, as well.

The principles that account these capabilities have also created some unexpected problems. With the increased demand to move application and tasks from mainframes and onto client-server based networks, the typical user is often faced with designing and implementing his own database-oriented applications. Although several data management systems simplify numerous implementation considerations, the approach to designing a usable database application is not particularly easy to understand by most users.

Typically a user starts out with an intuition about what kind of data he or she needs and where it can be found, and how the data should be presented. Then, however, he or she is

faced with tearing the problem apart-figuring out a set of tables--and then re-synthesizing his or her intuition, for producing queries, designing multi-table forms and reports, etc. In the past, a user typically interacted with a database management system (e.g., like an Oracle database) through the use of structured query language ("SQL") statements that logically define the parameters of the search query.

However, for a user to accurately query a database with such statements or commands, the user needed a working knowledge of the particular query language and also had to be intimately familiar with the structure of the particular database itself. Unfortunately, the task was complicated even more by the fact that structured query languages are perceived to be too complex for most users to comprehend and use efficiently.

To shield a user from the complexities of a particular query language, many database systems employed a menu system with a set of predefined commands or query statements that an end user could use to access the database. More recently, database systems developers have added a graphical user interface ("GUI") that allows an end user to easily access and query a database without the knowledge of specific, structured query language. For example, U.S. Pat. No. 5,555,403, issued to Cambot et al. for a Relational Database Access System Using Semantically Dynamic Objects, describes a system that allows an ordinary user to query a database without knowing either the structure of the database or any particular structured query language. The system appears to use an interface that allows users to define database queries by using a familiar a "point and click" interface. The system then translates the graphical query into a structured query command that the relational database system can understand. Once the database management system executes the translated query, the user interface displays the results of the query in a tabular form.

As represented by the above reference, the integration of a relational database management system with a graphical user interface greatly enhanced and facilitated the ease with which a database system could be accessed or modified. But these integrated systems were typically developed by software professionals who were familiar with the structure of the database and the language used to query the database. As a result, each database application required a customized graphical user interface that was specifically configured and tailored to meet the unique requirements for accessing and presenting data from the database. Additionally, many existing database applications evolved over time and thus required changes to the underlying database structure. Thus, for each modification to the structure of the database (e.g., adding one or more columns to the database or modifying the attributes of existing columns), a corresponding change to the customized graphical user interface was generally required, as well. Once again, these changes required the expertise of a database administrator or software developer, and was time consuming and costly, particularly where the database application was used as a management tool for dynamic and growing commercial enterprise. What users really want is not to deal with a set of disciplined tables but, instead, forms and reports that users have envisioned in their minds. Moreover, the forms and reports desired by users often contain nested structures (e.g., multi-table reports), yet users do not want to be concerned with the nested structures themselves, such as how to link or fuse information from a number of database sources.

What is needed is a new approach for a user to develop an application. In particular, the user should not be required to understand or to even know about low level details of database applications development for accessing, retrieving and displaying data, linking tables via foreign keys, defining key fields, etc. Since a user already has in his or her mind a good idea of the form or report desired, and how it should look

on a video display in real time; such a system should provide an intuitive, visual paradigm for creating the application. In this way, the user focuses on objects that are very visual in nature and easily understood, thereby providing a better match with what the user wants to do.

It is with respect to these and other considerations, limitations and problems, that the technique of the present invention has evolved.

SUMMARY OF THE INVENTION

In accordance with the present invention, a computer implemented "DATA INTEGRATOR" system for obtaining, storing and displaying cells of visual information such as Internet World Wide Web pages includes a display unit, a storage system for storing cells in eXtensible Markup Language (XML) format, and a communication unit for obtaining data objects or documents which can be in a plurality of different formats from the Internet and other external sources.

A cell designer enables a user to create a view including a plurality of selected cells, each of which includes at least one data object. A display generator obtains the selected data objects from the storage system and generates the view for display by the display unit. A conversion unit converts data objects obtained by the communication unit into XML using eXtensible Style Language (XSL) and Document Type Definitions (DTD), and the cell designer creates the specified arrangement using XSL and DTD.

The conversion unit includes a plurality of factories for converting data objects obtained by the communication unit from different respective formats into XML, and a factory manager for determining a format of a data object obtained by the communication unit and assigning it to a corresponding factory.

An event manager causes the communication unit to obtain updated versions of outdated data objects from the external sources. A list unit stores a list of users and data objects

requested by each user, and the event manager sends notifications of the updated versions to users who have requested the specified data objects.

A preferred embodiment of the present invention provides a method and apparatus for the collecting, fusing and altering the structure of information for presentation. The present invention offers significant advantages and improvements that achieve a true dynamic interface for use with both relational or object-oriented database management systems and web-based data content providers alike. The system allows users to search and edit the contents of a database. The present invention further provides an efficient method of importing and exporting information in the form of XML based documents to and from these databases.

In accordance with these general aspects, the retrieval and display system of the present invention is executed by programmed instructions of a general-purpose computer in response to input information. The general-purpose computer includes a memory in which the programmed instructions are recorded, an input device to supply input information for interaction with the programmed instructions, and a display device for displaying information created by the programmed instructions and the input information. The functions achieved by the programmed instructions involve creating a dynamic, self-modifying display processor. The display processor allows a user to search and edit the contents of a database using a plurality of palettes or panels. This allows a user to edit the data content of the database without requiring the use of a structured query language.

Furthermore, the present invention is responsive to user demands by modifying and regenerating the windows or cells of the user-created display each time data is changed, so that the windows or cells accurately reflect the current data content of the database. Lastly, the display processor of the present invention further provides the capability to both edit the

content and modify the structure of information imported from other data sources.

In accordance with the purpose of the invention, as embodied and broadly described herein, the invention relates to creating and storing complex data structures or content in uniform text-based, platform and application independent mechanisms used to represent, store and exchange information with a variety of database sources. This allows a user to search and edit the contents of local, network-based and web-based data service providers without requiring the users to be familiar with relatively complex structured query languages.

In further accordance with the purpose of the invention, as embodied and broadly described herein, the invention relates to the ability that allows a user to create display components, custom views, and keep the information displayed current and up to date. Because the system is able to recognize changes in the data content of an XML based document, it is able to incorporate those changes in both the database and the display elements of the window or cell, without rendering the display interface develop by the user inoperative.

In accordance with these improved aspects, the present invention may be used in dynamic database environments where both the contents as well as the structure of the database are frequently changing. Such dynamic database environments would present severe challenges to other system because the use of a static retrieval and display would be too slow and too costly to accommodate time dependent or critical data required by a user.

These and other features and advantages of the present invention will be apparent to those skilled in the art from the following detailed description, taken together with the accompanying drawings, in which like reference numerals refer to like parts.

DESCRIPTION OF THE DRAWINGS

The accompanying drawings, which are incorporated in and constitute a part of this specification, illustrate several embodiments of the invention and, together with the description, serve to explain the principles of the invention.

FIG. 1 is a block diagram illustrating a system in which the present invention may be implemented.

FIG. 2 is a block diagram showing the primary operational elements of the present invention.

FIG. 3 is a block diagram illustrating the operative elements of a DATA INTEGRATOR Cell Designer 212 in detail.

FIG. 4 is a block diagram showing the functional elements of an XML Data Manager 210 in greater detail.

FIG. 5 is a block diagram illustrating the program elements of a PersistenceX Data Server 412 in greater detail.

FIGs. 6a and 6b in combination constitute a flowchart illustrating the steps performed by an XML Data Server 310 to service a client request in accordance with FIG. 4.

FIGs. 7a and 7b in combination constitute a flowchart showing the steps performed by an XML Data Server 310 to save an updated or new XML based document and update the Cache Manager.

FIG. 8 is a diagram illustrating a display created by the present invention.

FIGs. 9a to 9i are diagrams illustrating a cell designer wizard of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

The present invention now will be described more fully with reference to the accompanying drawings, in which the preferred embodiments of the invention are shown. The present invention may, however, be embodied in many different forms and should not be construed as limited to the embodiments set forth herein; rather these embodiments are provided so that this disclosure will be thorough and complete and will fully convey

the invention to those skilled in the art.

FIG. 1 is a block diagram illustrating a "DATA INTEGRATOR" system 100 embodying the present invention. The system 100 shown in FIG. 1 includes a Central Processing Unit 102 (or any appropriate processor or processors), random-access memory (RAM) 104, and some form of mass storage 106 (e.g., hard or fixed disk). The mass storage 106 element of system 100 is capable of storing system software 118, application programs 120, an array of databases 122, and in particular, the operating software and development tools of the present invention. The system 100 also includes an input/output controller 108, a keyboard 112, a display device 114, and a pointing device 116 (e.g., mouse or track ball). The system 100 may also be provided with additional input/output devices, such as a printer, if desired. The various components of the system 100 communicate through a system bus 110 or similar architecture, as shown.

In the following discussion, it is understood that an appropriate processor 102 (or similar processor) performs the steps of methods and flowcharts discussed preferably executing program instructions or code stored in mass storage 106. It will also be understood that the invention is not limited to any particular implementation or programming technique and that the invention may be implemented using any appropriate techniques for implementing the functionality described herein. The invention is not limited to any particular programming language or operating system.

While the present invention describes a single client application environment, it will be appreciated by those skilled in the art that the benefits and advantages of present invention are also applicable to a multiple client application environment as well.

FIG. 8 illustrates a display or "view" 10 which is created by the present system and displayed on the display device 114. As illustrated, the view appears in a standard window, and

consists of a number of visual information blocks called "cells". The data in the cells can be obtained from a variety of sources, such as a relational database and/or the Internet. In the latter case, the cells will typically be created from full or partial World Wide Web (WWW) pages consisting of HTML tags and content.

The present invention enables a user to create a view including a plurality of selected cells for example, a list of incidents 12 and a Decon Chart 14 for a disaster control command post as shown in FIG. 8. Each cell can include a plurality of data objects or elements. For example, the cell 12 includes elements "4th and Elm", "Chlorine Spill" and "All incidents".

As will be described in detail below, the present invention enables the user to specify which cells are to be included in the view, the size and location of each cell in the view, and which elements in each cell are to be included, as well as attributes of the cells and elements. The attributes include, for example, a type of display (e.g. text or graph), text font and size, and color.

FIG. 2 is a block diagram showing the primary operational units of the present Invention. These units can include computer hardware, software or a combination thereof.

As shown in FIG. 2, a consumer of information or client application 202 communicates with both a DATA INTEGRATOR Data Server 220 and a Cell Designer 212 of a DATA INTEGRATOR 200 via an application program interface called a DATA INTEGRATOR Data Bus 204. The DATA INTEGRATOR Data Bus 204 provides an easy exchange of data content and messages between the client application 202 and the DATA INTEGRATOR Data Server 220 without concern as to protocol or data format. The DATA INTEGRATOR Data Server 220 is a set of program modules that transforms and distributes data content from a plurality of data content providers and databases in a manner that can be easily displayed and manipulated by a client using the Cell Designer

212.

In addition, a person of ordinary skill in the art will understand that the system that runs the client application 202 is configured in a similar manner as system 100 described above. The storage system that runs the client application 202 stores system software, computer programs, platform-independent software, such as a user-oriented design tool like the Cell Designer 212, that allow the client application 202 to communicate with one or more of the databases or web-based data content providers via the DATA INTEGRATOR Data Server 220.

As also shown in FIG. 2, the DATA INTEGRATOR Data Server 220 includes an Extensible Markup Language (XML) Producer 206, an HTTP Connection Manager 208 which is connected to the Internet 209, and an XML Data Manager 210. The XML Producer 206 manages request traffic between the client application 202 and the XML Data Manager 210. The XML Producer 206 acts as an arbitrator receiving all messages and requests for data content placed on the DATA INTEGRATOR Data Bus 204 and forwards them to the XML Data Manager 210.

Any messages or requests directed to the XML Data Manager 210 are first sent through the HTTP Connection Manager 208 to resolve any underlying connection and protocol problems that might have been transmitted with the request across the DATA INTEGRATOR Data Bus 204 to the XML Data Manager 210. Once all the protocol and connection issues have been resolved, the HTTP Connection Manager 208 forwards the request to the XML Data Manager 210.

The XML Data Manager 210 retrieves the requested data objects from one or more of the available data content providers (e.g. World Wide Web (WWW) sites on the Internet) or databases and formats the data objects into a flexible, source independent XML format that are transmitted back to the client application 202 and then on to the Cell Designer 212. The Cell Designer 212 is used by the client application 202 to present and display XML based data content generated from the DATA

INTEGRATOR Data Server 220.

Reference is now made to FIG. 3 which is a block diagram illustrating the operative elements of Cell Designer 212 in detail. The Cell Designer 212 allows users to design a variety of custom presentations or displays of XML based cells. The cells are created by a dynamic user interface and a series of "attribute" buttons that direct the user to navigate through a series of presentation or display panels to design a new cell. The presentation or display panels in the Cell Designer 212 prompt the user for information related to the data content required and the manner in which it should be presented or displayed.

As FIG. 3 shows, the Cell Designer 212 includes an Application Interface 302, a Cell Descriptor 304 and a Bean Property Manager 306. Upon access to the Cell Designer 212, the user chooses the type of XML based document from a List of Available Documents Types 316 found in the Application Interface 302. The Application Interface 302 forwards the selected appropriate Document Type Definition (DTD) or XML Schema to the Cell Descriptor 304. The Document Type Definition (DTD) defines what data objects or elements are available to create the selected cell and how those elements are related.

The Cell Descriptor 304 then provides the user with a series of design palettes or panels to define specific elements to be displayed, such as the type of data to be displayed, what elements to display, what fields to be used to calculate data and the like.

As FIG. 3 also shows, the Cell Descriptor 304 comprises an Element Property Handler 308 and a Type Property Handler 310. Based on the document selected, the Type Property Handler 310 provides the user with a set of palettes or panels 314 that define display type and display properties. Display Type allows the user to choose the manner in which the selected data can be presented (i.e. chart, table or tree). The display

properties palette or panel 314 is used to refine the property elements of the display type chosen, and the Element Property Handler 308 provides a plurality of design palettes or panels 312 that guide the user in refining the information required and the manner in which it will be displayed.

By using the design palettes or panels, the user is able to define specific properties of individual display elements as well as stipulating preferences in the overall appearance of the cell.

The cell designer 212 includes a Graphical User Interface (GUI) which is illustrated in FIGs. 9a to 9i. FIG. 9a illustrates an initial screen which enables a user to initiate a cell designer session from an application menu. FIG. 9b illustrates how the cell designer 212 provides the option to create a new cell or retreive a previously designed cell. FIG. 9c shows how the cell designer 212 provides a list of available XML documents (data objects or elements) from the DTD list, and allows the user to select which document a cell will be created for.

FIG. 9d illustrates how the cell designer 212 allows the user to select which elements from the document to be included in the new cell. FIG. 9e shows how the cell designer is used to edit properties or attributes for each element in the new cell. FIG. 9f illustrates how the cell designer allows the user to select the type of cell to be created.

FIG. 9g illustrates how the cell designer enables the user to edit properties for the selected cell type. The display properties are based on the type of cell being created.

FIG. 9h shows how the cell designer 212 allows the user to define additional properties for the cell. These properties can be used by the application for such things as providing lists of available cells. FIG. 9i illustrates how a single cell can be displayed in an application.

FIG. 4 is a block diagram showing the functional elements of the XML Data Manager 210 in greater detail. FIG. 4 shows an

XML Servlet 404 or communication unit which receives all incoming requests from the HTTP Connection Manger 208. The HTTP Connection Manager 208 uses a GET request to enable the XML Servlet 404 and access the XML Data Manager 210. A Human Resources (HR) director 402 is a list unit which stores a list of users and documents requested by each user. The XML Servlet 404 interrogates each incoming request to determine the identity of the client and if it can be associated with one or more of the documents maintained by the HR Director 402.

The list or profile maintained by the HR Director 402 includes the identity of the client and a number of useful client statistics including type, structure and format of previously requested data or information. Thus, if a match exists, the XML Servlet 404 then transfers the matching profile object with the request to a Factory Manager 406.

As FIG. 4 shows, the Factory Manager 406 maintains and monitors a plurality of data content collection and conversion units called Factories. Each Factory 414 to 418 is designed to interpret and understand the nature of a specific request, as well as the manner in which the data is retrieved from a data content provider or database. The Factory Manager 406 analyzes the request from the XML Servlet 404 and determines which Factory 414 to 418 is able to efficiently service the request. The Factory Manager 406 then notifies the XML Servlet 404 as to which Factory 414 to 418 has been assigned to service the request.

In response, the XML Servlet 404 then passes the request for the data content in the form of an XML based document to the designated Factory 414 to 418 for processing. An Extensible Markup Language (XML) based document or data object is a text-based, platform and application independent mechanism used to represent, store and exchange specified data content.

FIG. 4 also shows the XML Data Manager 210 as including a PersistenceX Data Server 412, an Event Manager 408, and a Cache Manager 410. The PersistenceX Data Server 412 manages the

interface between the attached, resident database 420 and the XML Data Manager 210. As FIG. 4 shows, each Factory 414 to 418 submits its request for a XML based document and a copy of an Extensible Stylesheet Language (XSL) based map or template to the PersistenceX Data Server 412 for processing.

The PersistenceX Data Server 412 uses the XSL based template or map to formulate the structure and appearance of the retrieved data content in an XML based document required by Factory 414 to 418 to service the client request.

The Cache Manager 410 maintains a substantial library of XML based documents that are frequently accessed and previously requested. Each time an XML based document is updated or created from the database and then used, it is placed in the Cache Manager 410 to service future requests. This allows a Factory 414 to 418 to query the library of XML based documents maintained by the Cache Manager 410 first before engaging the PersistenceX Data Server 412 to create a new XML based document to service the request.

If a current version of the requested XML based document is stored in cache and is available, the Cache Manager 410 retrieves the desired XML based document that satisfies the request and alerts the servicing Factory 414 to 418 that it forwarded the requested document to the XML Servlet 404. The XML Servlet 404 then transmits the request through the HTTP Connection Manager 308 and on to the client application 302.

If the document cannot be found or needs to be updated, the Cache Manager 410 returns a message to the Factory 414 to 418 responsible for serving the request. Upon receipt of the message, the Factory 414 to 418 then directs the PersistenceX Data Server 412 to generate an XML based document that meets the requirements of the request. The PersistenceX Database Server 412 performs a conversion operation by using the XSL based template to create a new XML based document from data supplied by either the attached resident database 420 or other data content providers.

Once the XML based document has been built, the PersistenceX Data Server 412 forwards the newly created XML based document to the Factory 414 to 418 responsible for serving the request. The Factory 414 to 418 then transfers the newly created XML based document to the Cache Manager 410. After storing the XML based document in its cache, the Cache Manager 410 then returns the requested document to the Factory 414 to 418 to be transmitted via the XML Servlet 404 and HTTP Connection Manager 308 to the client application 202 for presentation and display.

As FIG. 4 shows, the Event Manager 408 interfaces with both the Cache Manager 410 and the HR Director 402. The Event Manager 408 is a program module that monitors any change of events such as updates of the database or updates to XML based documents stored in cache, for example. If such changes occur, the Cache Manager 410 identifies what XML based documents have changed and communicates the status to the Event Manager 408. The Event Manager 408 then obtains the list of affected subscribing client applications from the HR Director 402 and notifies these subscribing client applications using an XML based document of the changes and to request an update.

FIG. 5 is a block diagram illustrating the program elements of the PersistenceX Data Server 412 in greater detail. As FIG. 5 shows the PersistenceX Data Server 412 comprises a PersistenceX Processor 502, a XSL Processor 504 and a Persistence Manager 506. The PersistenceX Processor 502 manages the process of retrieving data content from one or more local databases or web-based data content providers 420 and, in particular, creates the XML based documents in response to requests made by one of the Factories 414-418. The XSL Processor 504 generates a relational transform from the XSL based template and other associated parameters supplied to the PersistenceX Processor 502 by the servicing Factory 414 to 418. The PersistenceX Processor 502 uses these elements or parameters from the transform to query both the local databases

and web-based databases 420 for the required data content.

Each query is done through the Persistence Manager 506. The Persistence Manager 506 supplies the necessary connectivity to the PersistenceX Processor 502 in order to query one or more of the databases and retrieve the data content needed to create a satisfactory XML based document. The PersistenceX Processor 502 then generates an XML based document in accordance with the XSL based template or map and forwards it to the servicing Factory 414-818.

FIG. 6a is a flowchart illustrating the steps performed by an XML Data Server 310 to service a client request in accordance with FIG. 4. As FIG. 6a shows, the client application in step 602 connects to the XML Data Server 310 and a GET request is sent to initialize the interface. After connection, the request in step 604 is transferred from the client application. The content of the request in step 606 is analyzed to determine what Factory 414 to 418 in step 608 will service the request.

If in step 610 there is no Factory to service the request, it is returned to the client application for revision. But, as in step 612 if there is a Factory that can service the request, a confirmation is returned to the XML Servlet 404. In response, the request for an XML based document in step 614 is returned to the Factory for processing.

FIG. 6a also shows that the Factory in step 616 queries the Cache Manager 410 to determine if the requested XML based document is stored in cache. If the document in step 618 can be found in the cache, the document is retrieved and returned to the XML Servlet 404. The XML Servlet in step 620 forwards the XML based document to the client application 202.

On the other hand, if the XML based document cannot be found in cache, the PersistenceX Data Server 412 in step 622 is directed to generate a new XML based document based on an existing and stored XSL based template. The newly created XML based document in step 622 is then returned to the Factory and

is stored by the Cache Manager 410 in cache. Once again, the XML based document is retrieved in step 618 and returned to the XML Servlet 404 and the XML Servlet 404 in step 620 then returns the XML based document containing the desired data content of the client application 202. The PersistenceX Data Server 412 in step 622 is instructed to generate an XML based document.

FIG. 6b shows substeps of the step 624 performed by the PersistenceX Data Server 412 to retrieve and generate an XML based document in greater detail. As FIG. 6b shows, the PersistenceX Processor 412 in step 624a receives the XSL based template, the XML request and other associated parameters from the Factory 414 to 418 servicing the request. The PersistenceX Processor 412 in step 624b sends the XSL based template and the associated parameters to the XSL Processor 504. The XSL Processor 504 in step 624c generates a relational transform from the XSL based template, and the associated parameters. The PersistenceX Processor 412 uses the relational transform in step 624d to direct the Persistence Manager 506to establish the appropriate connections to the databases having the data required to service the client application request.

In addition, the PersistenceX Processor 412 in steps 624e and 624f uses the same relational transform to query these databases. Using the results from the database queries and the relational transform as a template, the PersistenceX Processor 412 in step 624g constructs a XML based document. The PersistenceX Processor 412 forwards XML based document to the Factory 414 to 418 that originated the request.

The Factory then transfers the XML document to the Cache Manager 410 that places the XML based document in the cache. Then, the Factory in step 620 returns it to the XML Servlet 404. The XML Servlet 404 returns the XML based document to the client application 202.

Figs. 7a and 7b in combination constitute a flowchart showing the steps performed by an XML Data Manager 210 to save

an updated or new XML based document and update the Cache Manager 410. The XML Data Manager 210 in step 702 receives a POST request indicating that the client application wishes to save an updated XML based document. The Factory Manager 406 in step 704 determines which Factory 414 to 418 is able to service the request and sends a confirmation of the selection to the XML Servlet 404. The XML Servlet 404 in step 706 receives and passes the XML based document to be saved to the Factory.

The Factory in step 708 retrieves the relational transform or template associated with the XML based document to be saved. The relational transform or template and the XML based document in step 710 are transferred to the PersistenceX Data Server 412. The PersistenceX Data Server 412 in step 712 uses the template to save the XML based document.

In addition, the PersistenceX Data Server 412 notifies the Factory 414 to 418 that it has saved the XML based document. The Factory in step 714 determines if the steps of saving the XML based document was valid. If the status in step 714 is not valid, the Cache Manager 410 in step 716 leaves the currently stored copy of the XML based document alone and sends a message in step 718 as to the status of the XML based document to the client application 202 via the HTTP Connection Manager 208.

FIG. 7b shows the steps performed by the PersistenceX Data Server 412 when saving the XML based document is determined to be valid. As shown, the Factory in step 720 alerts the Cache Manager 410 to invalidate and discard the currently stored XML based document. Once the Cache Manager 410 has discarded the XML based document, the Cache Manager 410 in step 722 notifies the Event Manager 408 of the change and returns control to the Factory.

The Factory in step 722 also returns an "OK status" to the client application 202. At the same time, the Event Manager 408 in step 724 obtains a list of subscribing client applications that were using the invalidated XML based document from the HR Director. After receiving the user list, the Event

Manager in step 726 uses HTTP Connection Manager 208 to notify the subscribing client applications of the change and to update their display.

In summary, the present invention overcomes the limitations of the prior art by providing a system which enables users to effectively obtain, store and display cells of data from a variety of external sources. Various modifications will become possible for those skilled in the art after receiving the teachings of the present disclosure without departing from the scope thereof.

WHAT IS CLAIMED IS:

1. A computer implemented system for obtaining, storing and displaying cells of information, comprising:

a display unit;

a storage system for storing data objects in a predetermined format;

a communication unit for obtaining data objects which can be in a plurality of different formats from an external source;

a conversion unit for converting data objects obtained by the communication unit into the predetermined format for storage in the storage system;

a cell designer which enables a user to create a view including a plurality of selected cells, each of which includes at least one data object; and

a display generator for obtaining the data objects for the selected cells from the storage system and generating the view for display by the display unit.

2. A computer implemented system as in claim 1, in which the predetermined format is eXtensible Markup Language (XML).

3. A computer implemented system as in claim 2, in which the cell designer creates the specified arrangement using eXtensible Style Language (XSL) and Document Type Definitions (DTD) associated with the specified cells respectively.

4. A computer implemented system as in claim 2, in which the conversion unit converts data objects obtained by the communication unit into XML using eXtensible Style Language (XSL) and Document Type Definitions (DTD) associated with the cells respectively.

5. A computer implemented system as in claim 1, in which the conversion unit comprises:

a plurality of factories for converting data objects obtained by the communication unit from different respective formats into the predetermined format; and

a factory manager for determining a format of a data object obtained by the communication unit and assigning it to a corresponding factory.

6. A computer implemented system as in claim 1, in which the storage system comprises a database for persistently storing cells and a cache for transiently storing cells.

7. A computer implemented system as in claim 6, in which the conversion unit is configured to transfer data objects between the cache and the database.

8. A computer implemented system as in claim 6, in which the display generator attempts to obtain a selected data object from the cache, and if it is not present, attempts to obtain it from the database.

9. A computer implemented system as in claim 1, further comprising an event manager for causing the communication unit to obtain an updated version of a specified data object from the external source, the updated version being converted into the predetermined format by the conversion unit, stored in the storage system and inserted into a corresponding displayed cell by the display generator.

10. A computer implemented system as in claim 9, further comprising a list unit for storing a list of users and data objects requested by each user, in which the event manager sends a notification of the updated version to users who have requested the specified data object.

11. A computer implemented system as in claim 9, in which the storage system comprises:

a database for persistently storing data objects;

a cache for transiently storing data objects; and

a cache manager for storing and invalidating data objects in the cache.

12. A computer implemented system as in claim 11, in which:

the cache manager notifies the event manager when it invalidates a data object; and

the event manager causes the communication unit to obtain an updated version of the invalidated data object from the external source in response to the notification.

13. A computer implemented system as in claim 1, in which the storage system is configured to cause the communication unit to obtain a selected data object from the external source if the selected data object is not stored in the storage unit.

14. A computer implemented system as in claim 1, in which the cell designer enables a user to arrange the cells in the view.

15. A computer implemented system as in claim 1, in which the cell designer enables a user to select attributes for the cells.

16. A computer implemented system as in claim 1, in which:

the cells can include pluralities of data objects; and

the cell designer enables a user to arrange the data objects in the cells.

17. A computer implemented system as in claim 1, in which:

the cells can include pluralities of data objects; and

the cell designer enables a user to create a view including a cell with less than all of its data objects.

18. A computer implemented system as in claim 1, in which:

the cells can include pluralities of data objects; and

the cell designer enables a user to select attributes for the data objects.

19. A computer implemented system as in claim 1, in which the communication unit, conversion unit, cell designer and display generator comprise computer software.

20. A computer implemented system as in claim 1, in which the external source comprises the Internet.

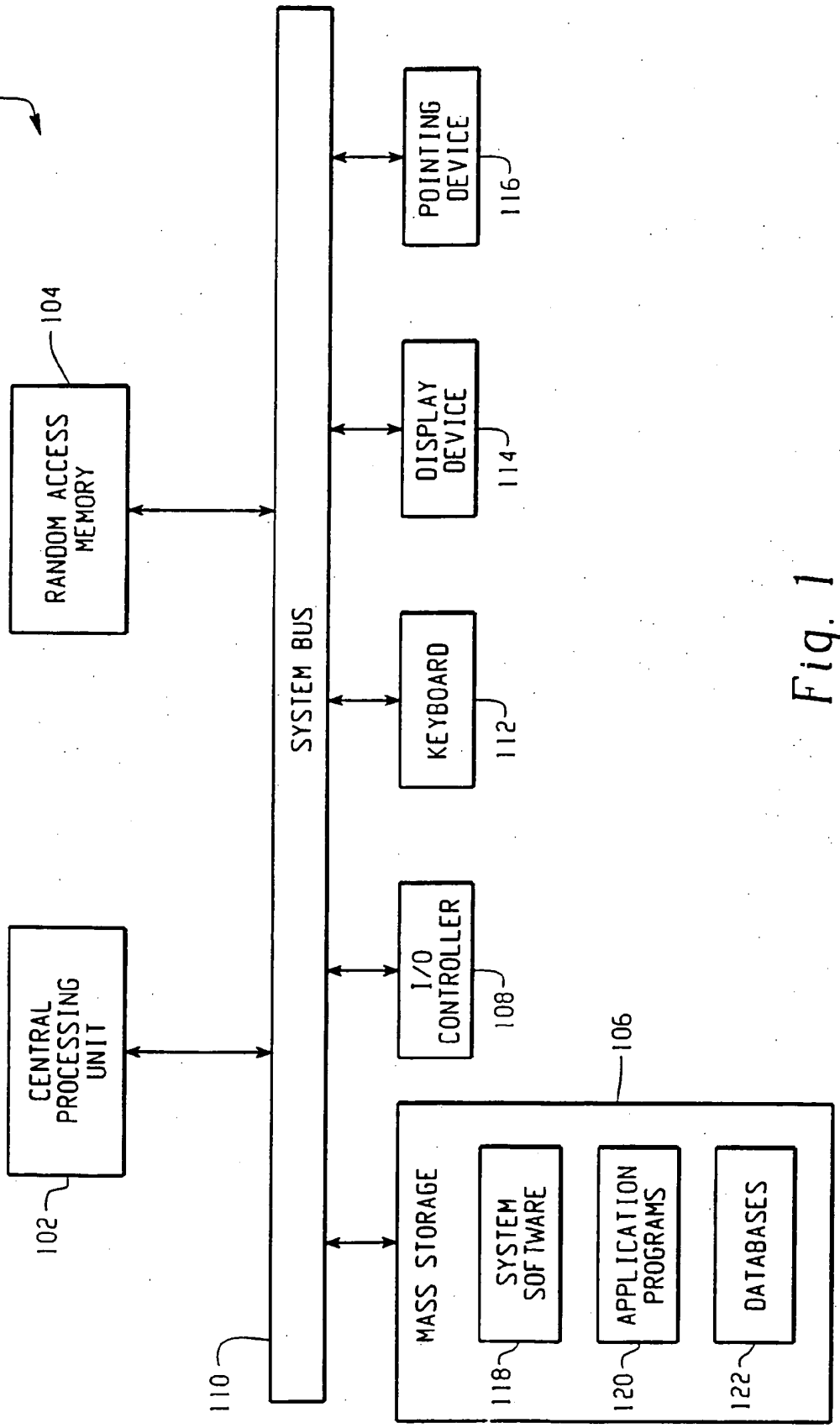

100
POINTING DEVICE
116
104
RANDOM ACCESS MEMORY
DISPLAY DEVICE
114
SYSTEM BUS
KEYBOARD
112
CENTRAL PROCESSING UNIT
I/O CONTROLLER
108
106
MASS STORAGE
SYSTEM SOFTWARE
APPLICATION PROGRAMS
DATABASES
102
110
118
120
122

Fig. 1

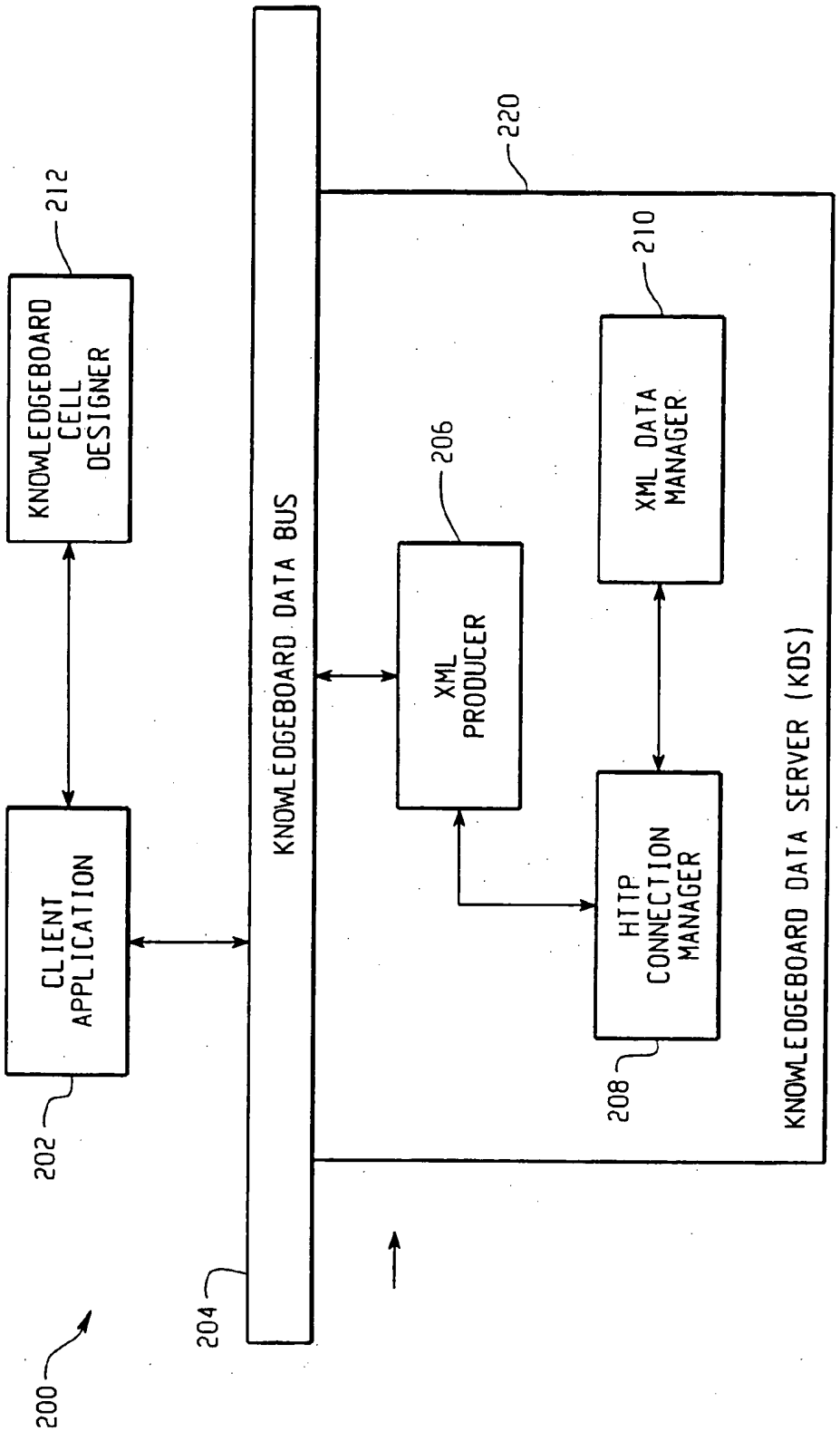

200
202
CLIENT APPLICATION
204
KNOWLEDGEBOARD DATA BUS
206
XML PRODUCER
208
HTTP CONNECTION MANAGER
210
XML DATA MANAGER
212
KNOWLEDGEBOARD CELL DESIGNER
220
KNOWLEDGEBOARD DATA SERVER (KDS)

Fig. 2

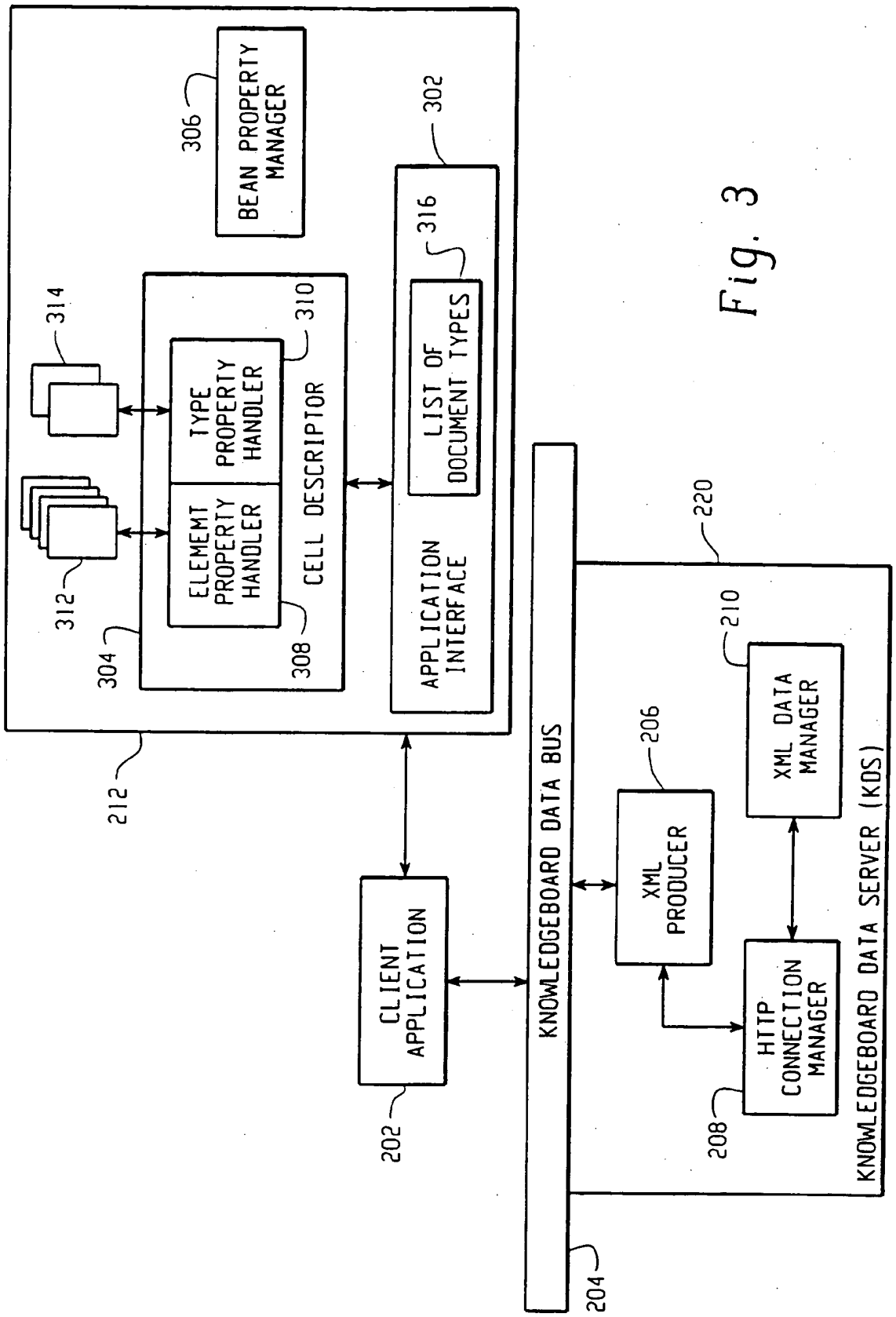

BEAN PROPERTY MANAGER
306
302
316
LIST OF DOCUMENT TYPES
314
310
TYPE PROPERTY HANDLER
CELL DESCRIPTOR
ELEMEMT PROPERTY HANDLER
APPLICATION INTERFACE
312
304
308
212
220
210
XML DATA MANAGER
206
XML PRODUCER
KNOWLEDGEBOARD DATA BUS
CLIENT APPLICATION
HTTP CONNECTION MANAGER
KNOWLEDGEBOARD DATA SERVER (KDS)
202
208
204

Fig. 3

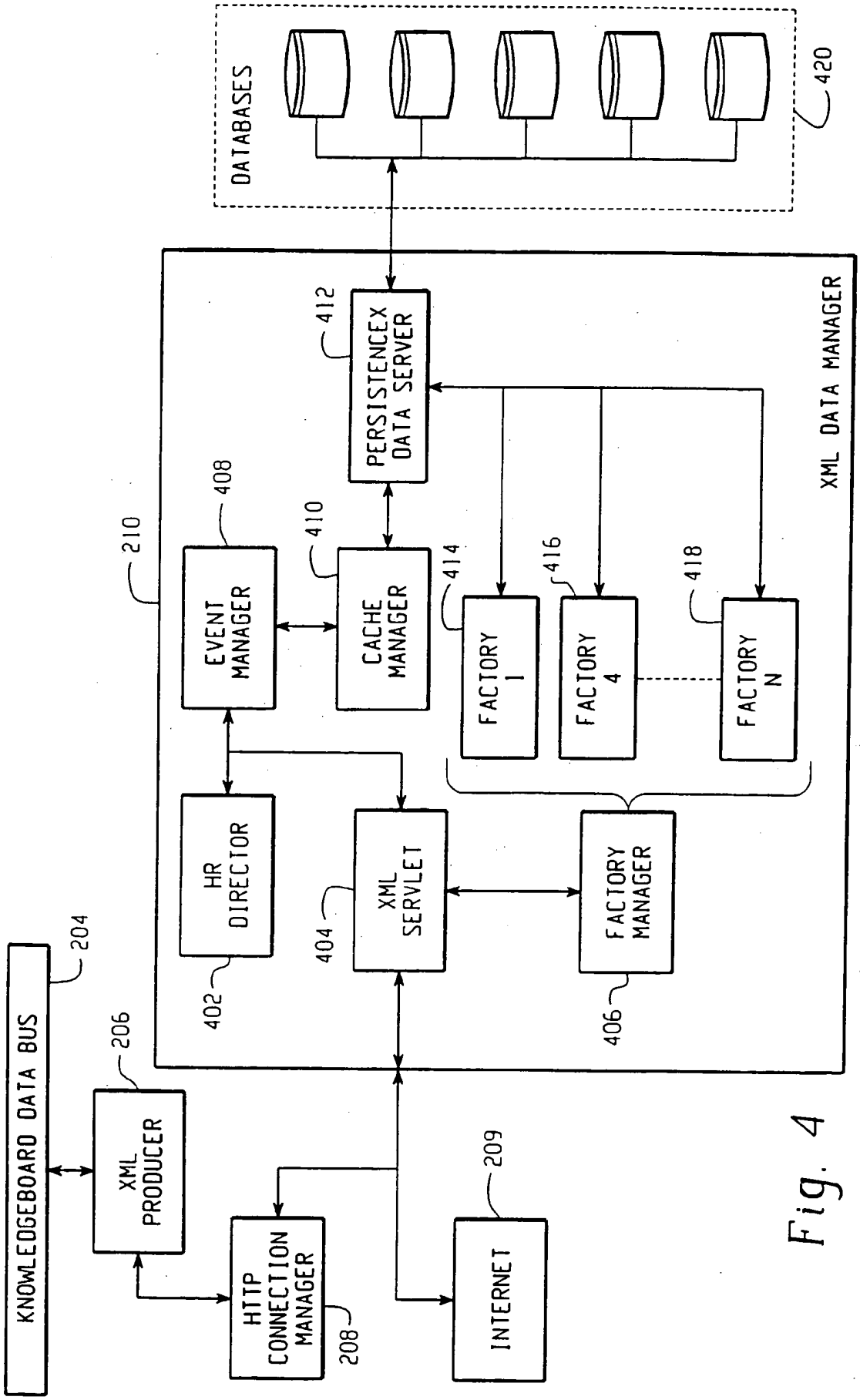

DATABASES
420
XML DATA MANAGER
PERSISTENCEX DATA SERVER
412
408
410
210
EVENT MANAGER
CACHE MANAGER
414
416
418
FACTORY 1
FACTORY 4
FACTORY N
HR DIRECTOR
XML SERVLET
FACTORY MANAGER
402
404
406
204
206
KNOWLEDGEBOARD DATA BUS
XML PRODUCER
HTTP CONNECTION MANAGER
208
INTERNET
209

Fig. 4

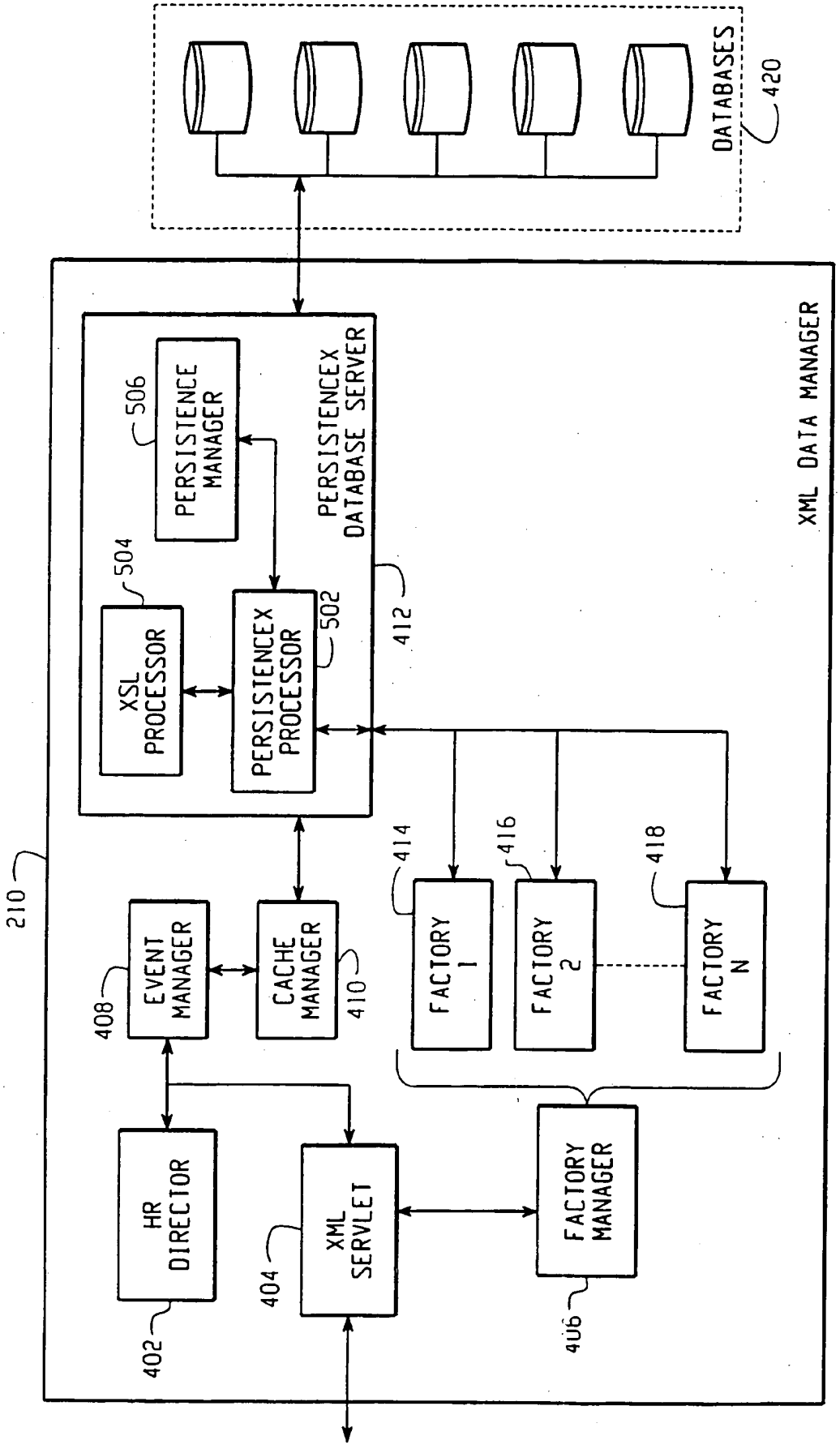

DATABASES
420
XML DATA MANAGER
210
PERSISTENCEX DATABASE SERVER
412
PERSISTENCE MANAGER
506
XSL PROCESSOR
504
PERSISTENCEX PROCESSOR
502
EVENT MANAGER
408
CACHE MANAGER
410
FACTORY 1
414
FACTORY 2
416
FACTORY N
418
HR DIRECTOR
402
XML SERVLET
404
FACTORY MANAGER
406

Fig. 5

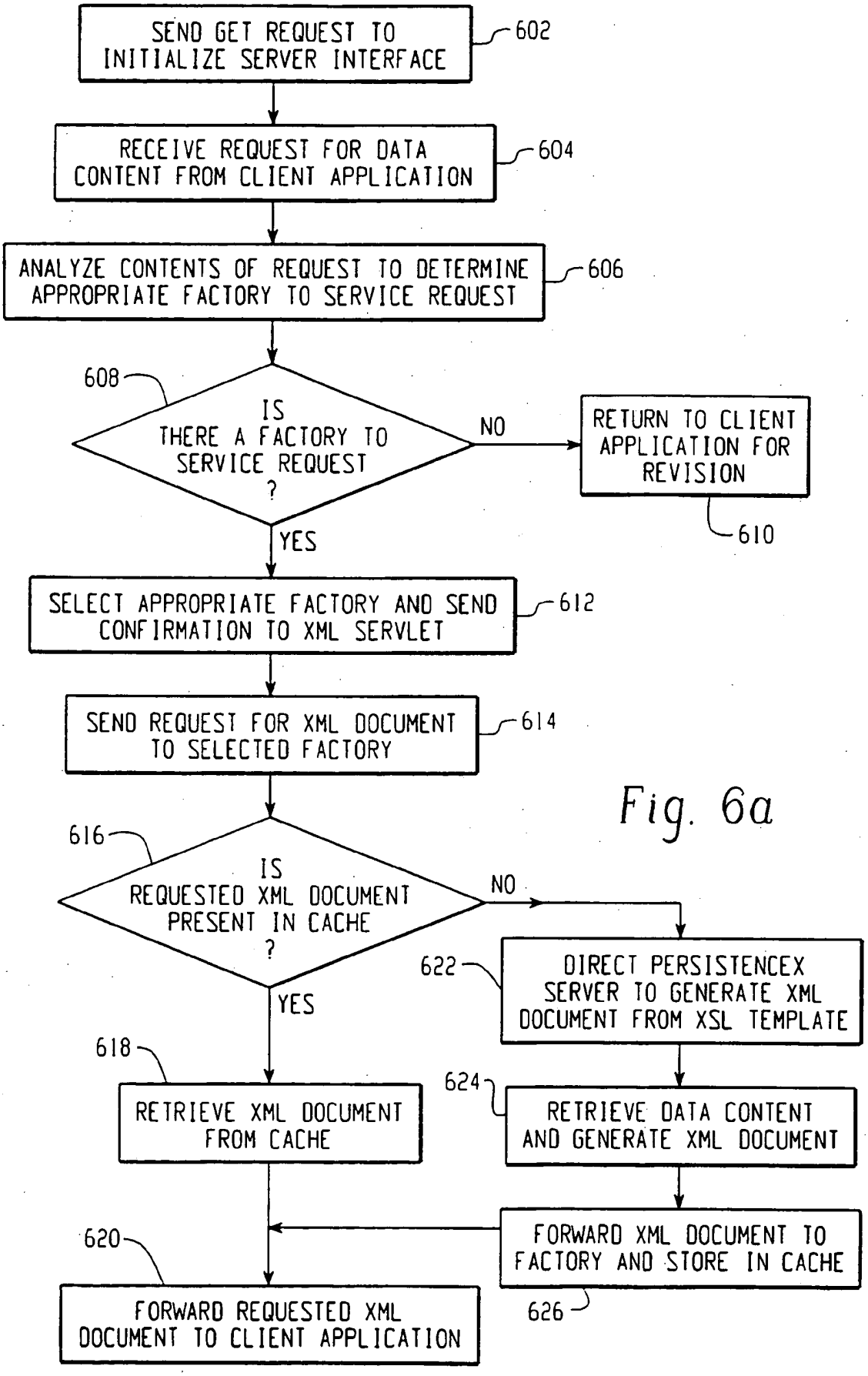

SEND GET REQUEST TO INITIALIZE SERVER INTERFACE
602
RECEIVE REQUEST FOR DATA CONTENT FROM CLIENT APPLICATION
604
ANALYZE CONTENTS OF REQUEST TO DETERMINE APPROPRIATE FACTORY TO SERVICE REQUEST
606
608
IS THERE A FACTORY TO SERVICE REQUEST ?
NO
RETURN TO CLIENT APPLICATION FOR REVISION
610
YES
SELECT APPROPRIATE FACTORY AND SEND CONFIRMATION TO XML SERVLET
612
SEND REQUEST FOR XML DOCUMENT TO SELECTED FACTORY
614
616
IS REQUESTED XML DOCUMENT PRESENT IN CACHE ?
NO
YES
622
DIRECT PERSISTENCEX SERVER TO GENERATE XML DOCUMENT FROM XSL TEMPLATE
618
RETRIEVE XML DOCUMENT FROM CACHE
624
RETRIEVE DATA CONTENT AND GENERATE XML DOCUMENT
FORWARD XML DOCUMENT TO FACTORY AND STORE IN CACHE
620
626
FORWARD REQUESTED XML DOCUMENT TO CLIENT APPLICATION

Fig. 6a

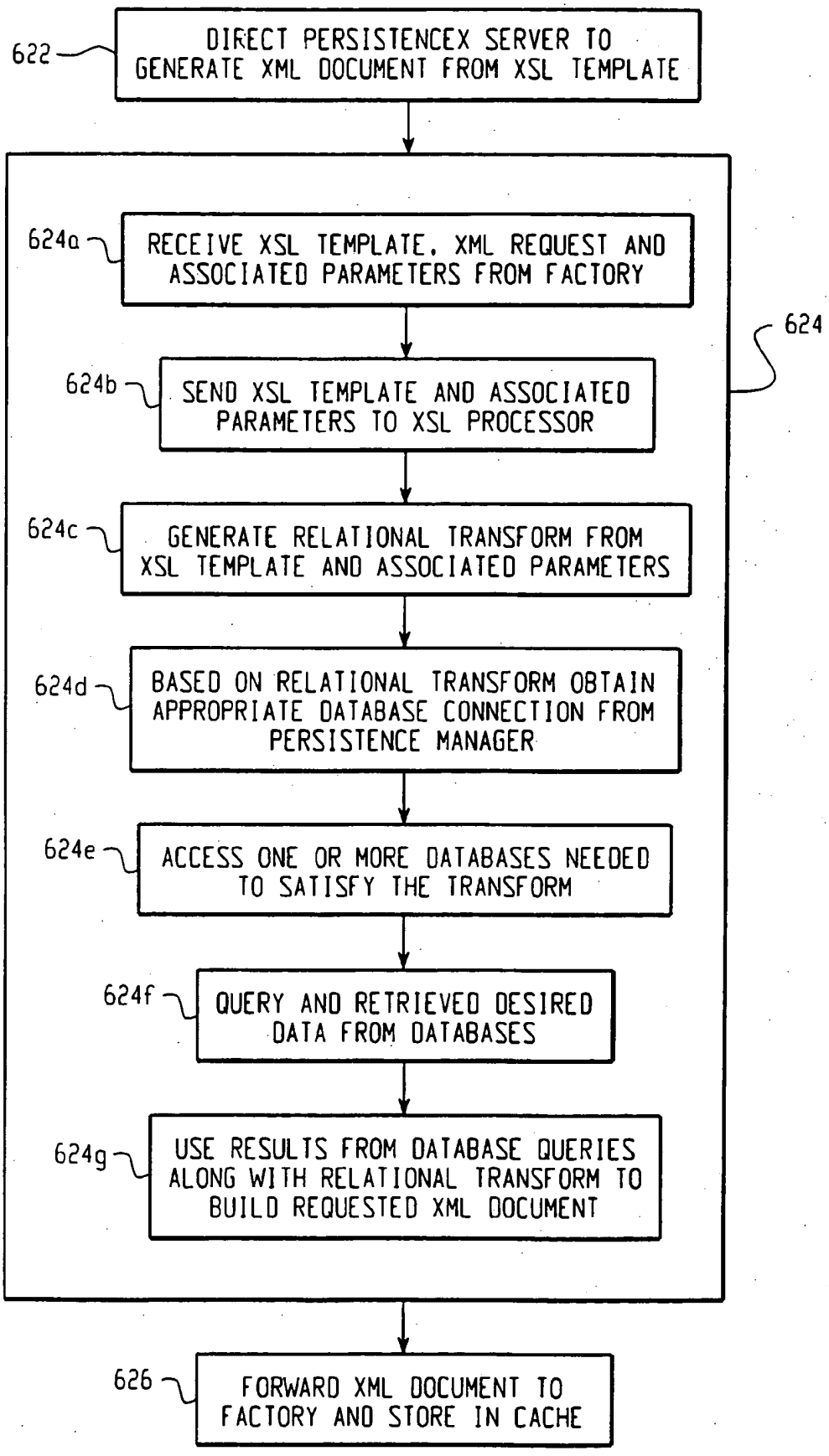

622
DIRECT PERSISTENCEX SERVER TO GENERATE XML DOCUMENT FROM XSL TEMPLATE
624
624a
RECEIVE XSL TEMPLATE, XML REQUEST AND ASSOCIATED PARAMETERS FROM FACTORY
624b
SEND XSL TEMPLATE AND ASSOCIATED PARAMETERS TO XSL PROCESSOR
624c
GENERATE RELATIONAL TRANSFORM FROM XSL TEMPLATE AND ASSOCIATED PARAMETERS
624d
BASED ON RELATIONAL TRANSFORM OBTAIN APPROPRIATE DATABASE CONNECTION FROM PERSISTENCE MANAGER
624e
ACCESS ONE OR MORE DATABASES NEEDED TO SATISFY THE TRANSFORM
624f
QUERY AND RETRIEVED DESIRED DATA FROM DATABASES
624g
USE RESULTS FROM DATABASE QUERIES ALONG WITH RELATIONAL TRANSFORM TO BUILD REQUESTED XML DOCUMENT
626
FORWARD XML DOCUMENT TO FACTORY AND STORE IN CACHE

Fig. 6b

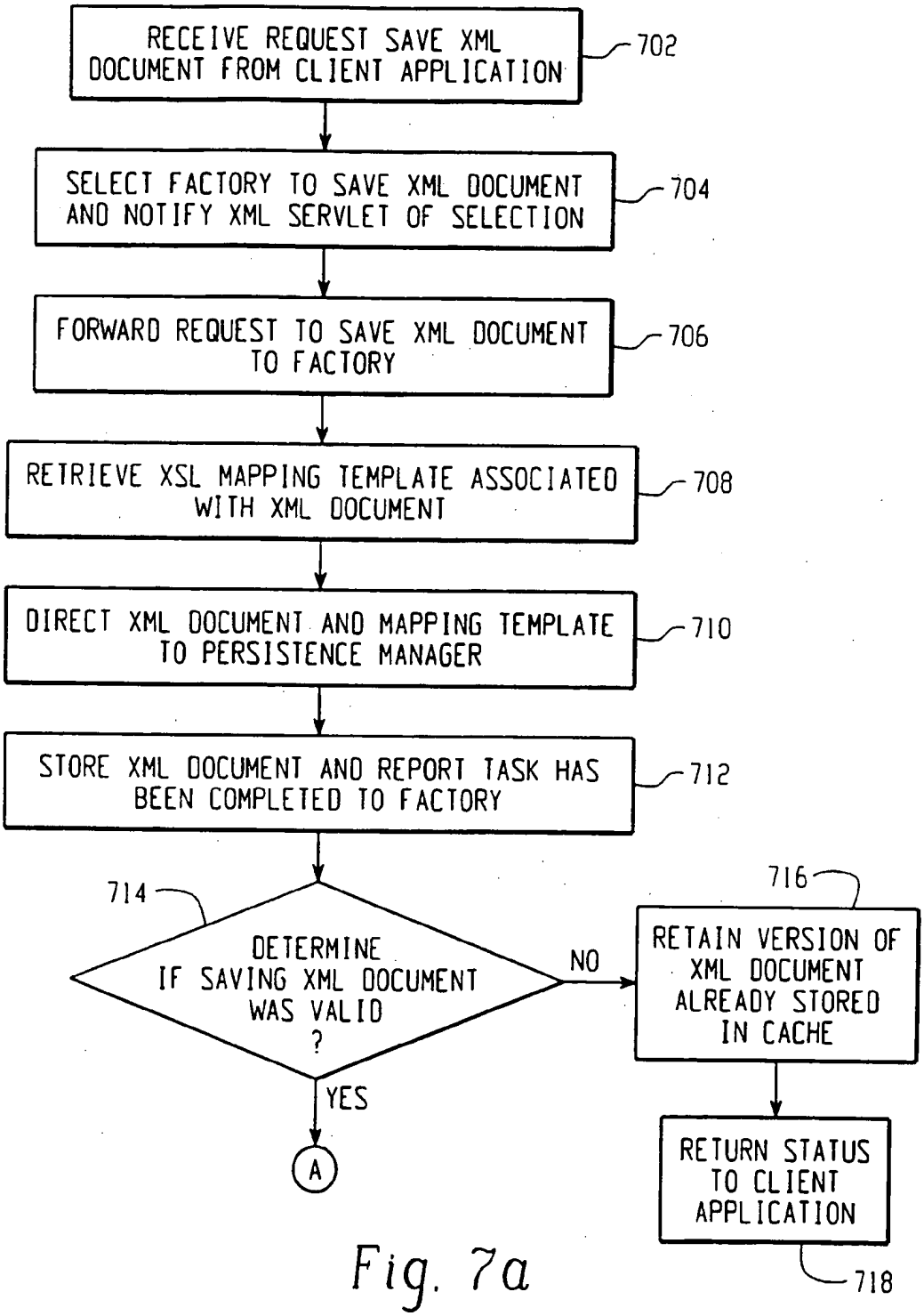

RECEIVE REQUEST SAVE XML DOCUMENT FROM CLIENT APPLICATION
702
SELECT FACTORY TO SAVE XML DOCUMENT AND NOTIFY XML SERVLET OF SELECTION
704
FORWARD REQUEST TO SAVE XML DOCUMENT TO FACTORY
706
RETRIEVE XSL MAPPING TEMPLATE ASSOCIATED WITH XML DOCUMENT
708
DIRECT XML DOCUMENT AND MAPPING TEMPLATE TO PERSISTENCE MANAGER
710
STORE XML DOCUMENT AND REPORT TASK HAS BEEN COMPLETED TO FACTORY
712
714
DETERMINE IF SAVING XML DOCUMENT WAS VALID ?
NO
716
RETAIN VERSION OF XML DOCUMENT ALREADY STORED IN CACHE
YES
A
RETURN STATUS TO CLIENT APPLICATION
718

Fig. 7a

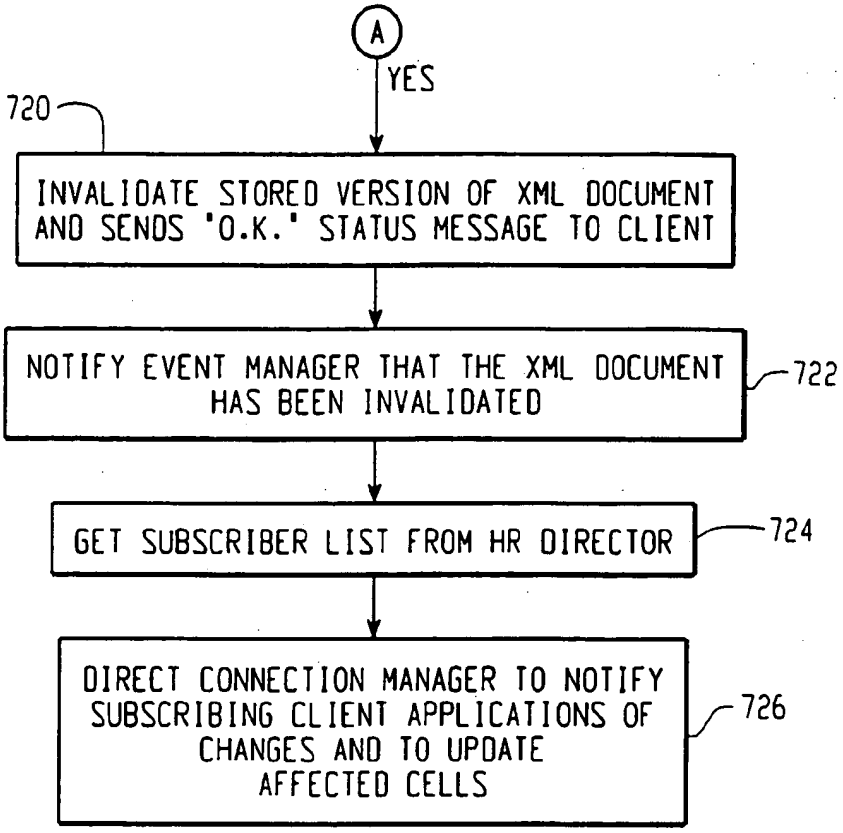

A
YES
720
INVALIDATE STORED VERSION OF XML DOCUMENT AND SENDS 'O.K.' STATUS MESSAGE TO CLIENT
NOTIFY EVENT MANAGER THAT THE XML DOCUMENT HAS BEEN INVALIDATED
722
GET SUBSCRIBER LIST FROM HR DIRECTOR
724
DIRECT CONNECTION MANAGER TO NOTIFY SUBSCRIBING CLIENT APPLICATIONS OF CHANGES AND TO UPDATE AFFECTED CELLS
726

Fig. 7b

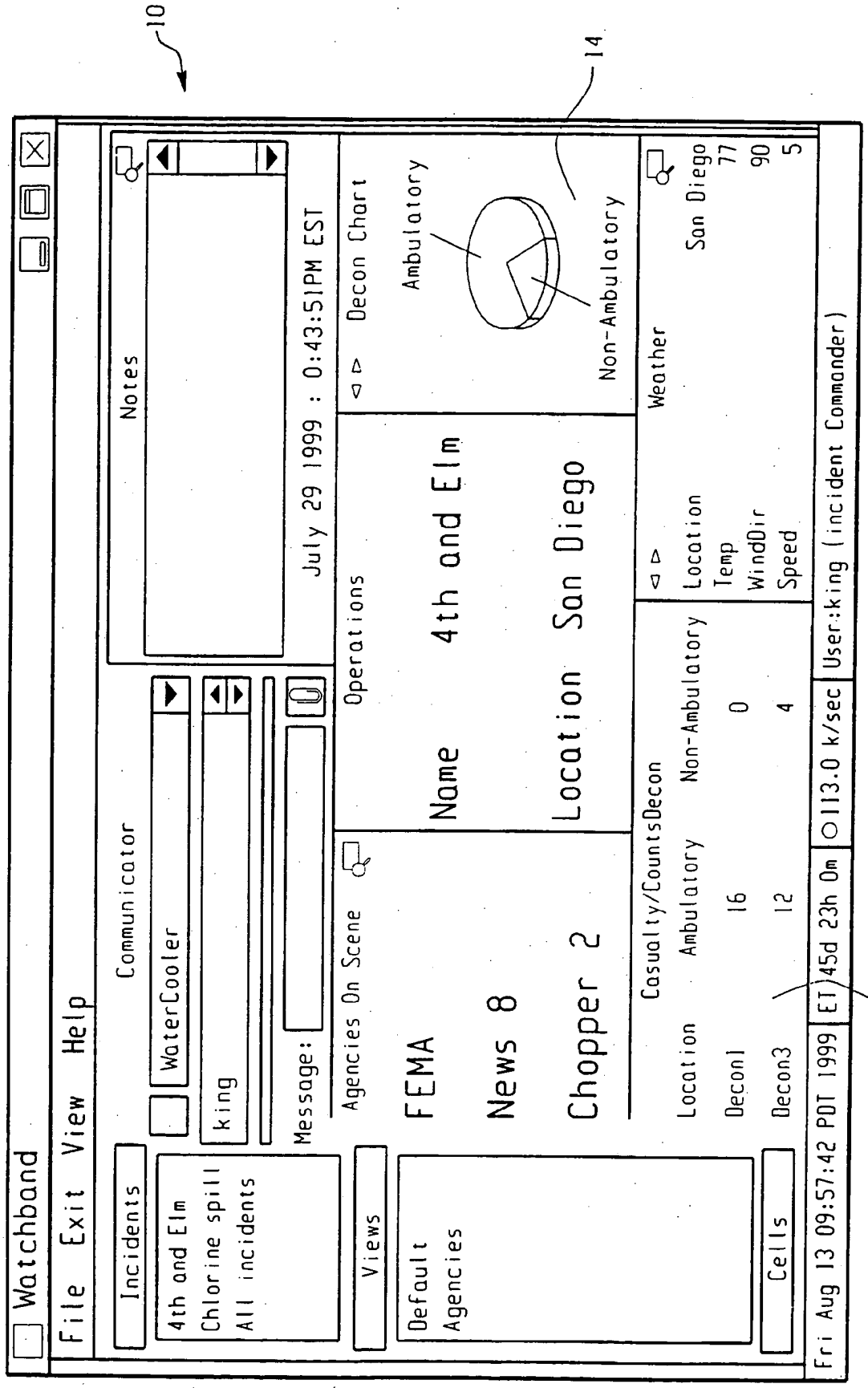

10
14
12
Watchband
File Exit View Help
Incidents
4th and Elm
Chlorine spill
All incidents
Views
Default
Agencies
Cells
Communicator
WaterCooler
king
Message:
Notes
July 29 1999 : 0:43:51PM EST
Agencies On Scene
FEMA
News 8
Chopper 2
Operations
Name 4th and Elm
Location San Diego
Casualty/CountsDecon
Location Ambulatory Non-Ambulatory
Decon1 16 0
Decon3 12 4
Decon Chart
Ambulatory
Non-Ambulatory
Weather
Location San Diego
Temp 77
WindDir 90
Speed 5
Fri Aug 13 09:57:42 PDT 1999
ET 45d 23h 0m
113.0 k/sec
User:king (incident Commander)

Fig. 8

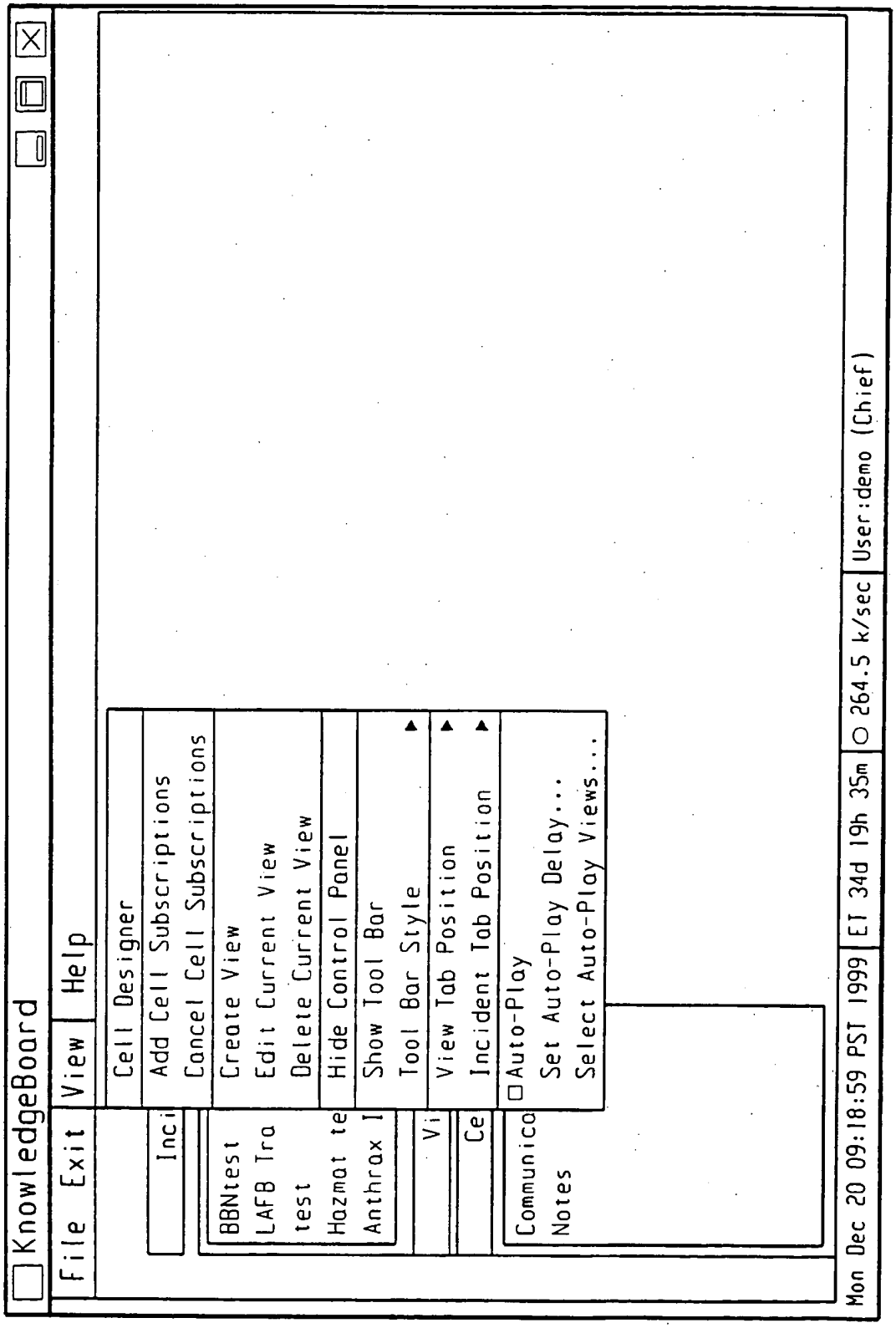

KnowledgeBoard
File Exit
View
Help
Cell Designer
Add Cell Subscriptions
Cancel Cell Subscriptions
Create View
Edit Current View
Delete Current View
Hide Control Panel
Show Tool Bar
Tool Bar Style
View Tab Position
Incident Tab Position
Auto-Play
Set Auto-Play Delay...
Select Auto-Play Views...
Inci
BBNtest
LAFB Tra
test
Hazmat te
Anthrax I
Vi
Ce
Communica
Notes
Mon Dec 20 09:18:59 PST 1999
ET 34d 19h 35m
264.5 k/sec
User:demo (Chief)

Fig. 9a

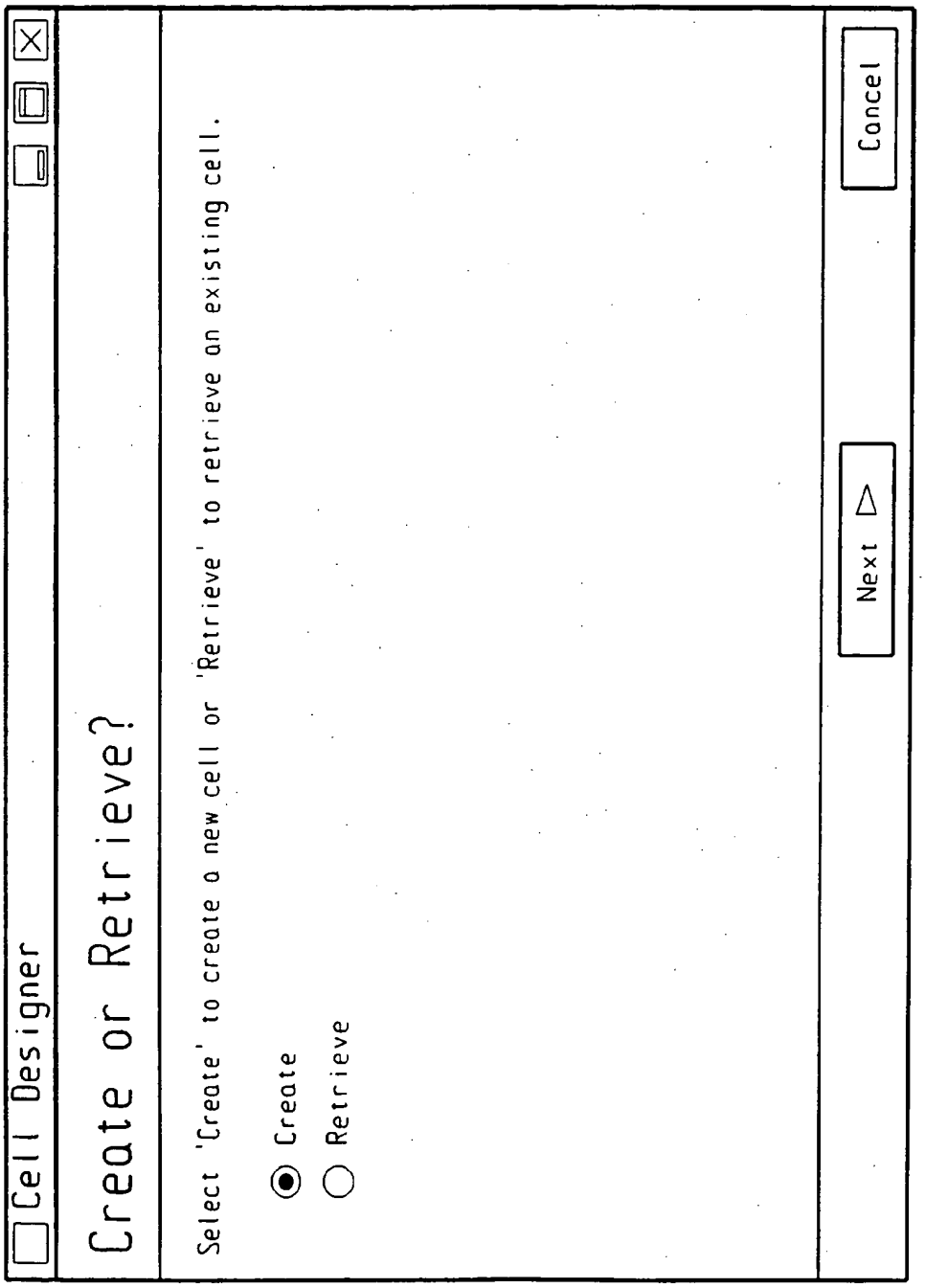

Cell Designer
Create or Retrieve?
Select 'Create' to create a new cell or 'Retrieve' to retrieve an existing cell.
Create
Retrieve
Next
Cancel

Fig. 9b

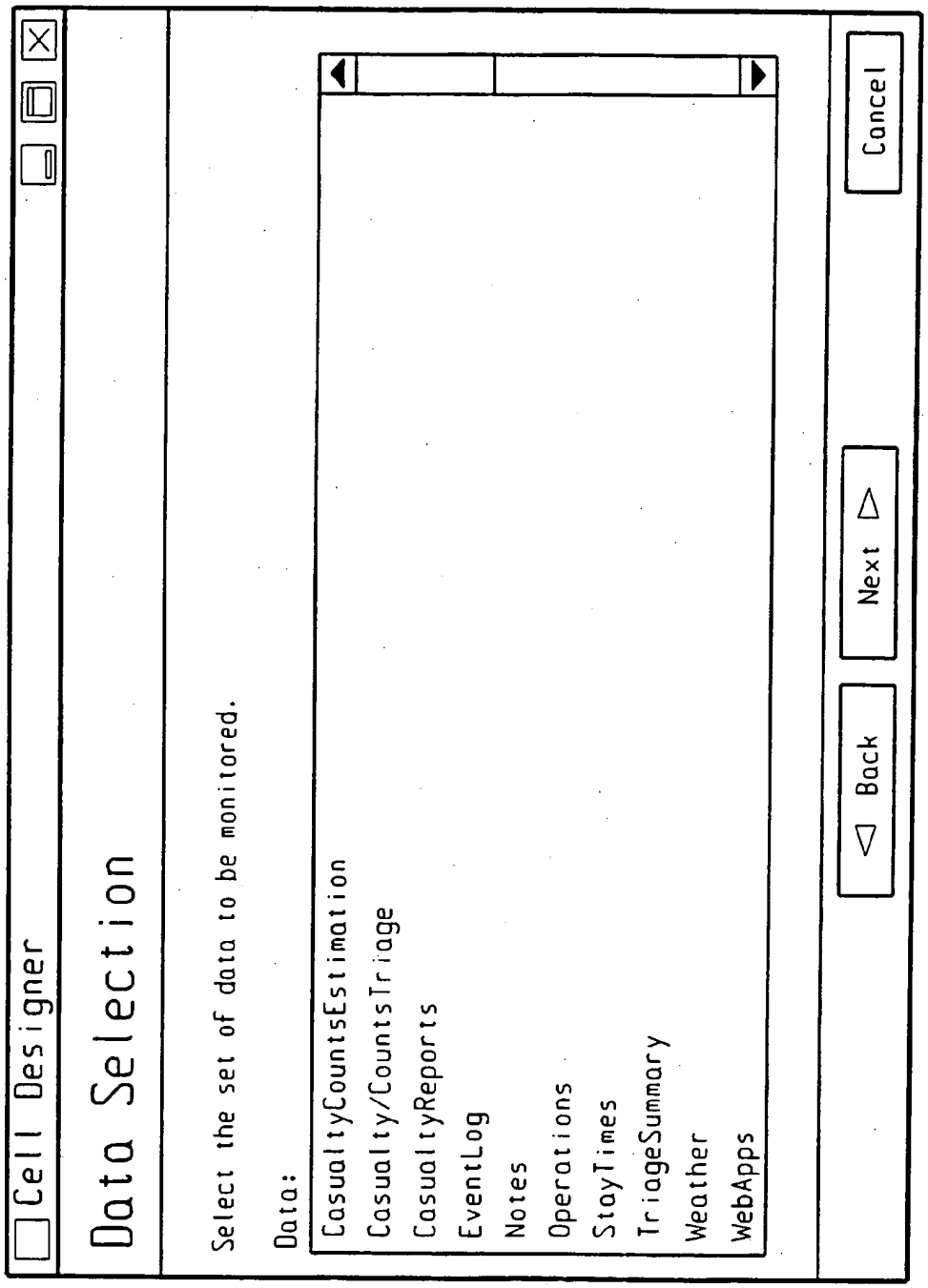

Cell Designer
Data Selection
Select the set of data to be monitored.
Data:
CasualtyCountsEstimation
Casualty/CountsTriage
CasualtyReports
EventLog
Notes
Operations
StayTimes
TriageSummary
Weather
WebApps
Back
Next
Cancel

Fig. 9c

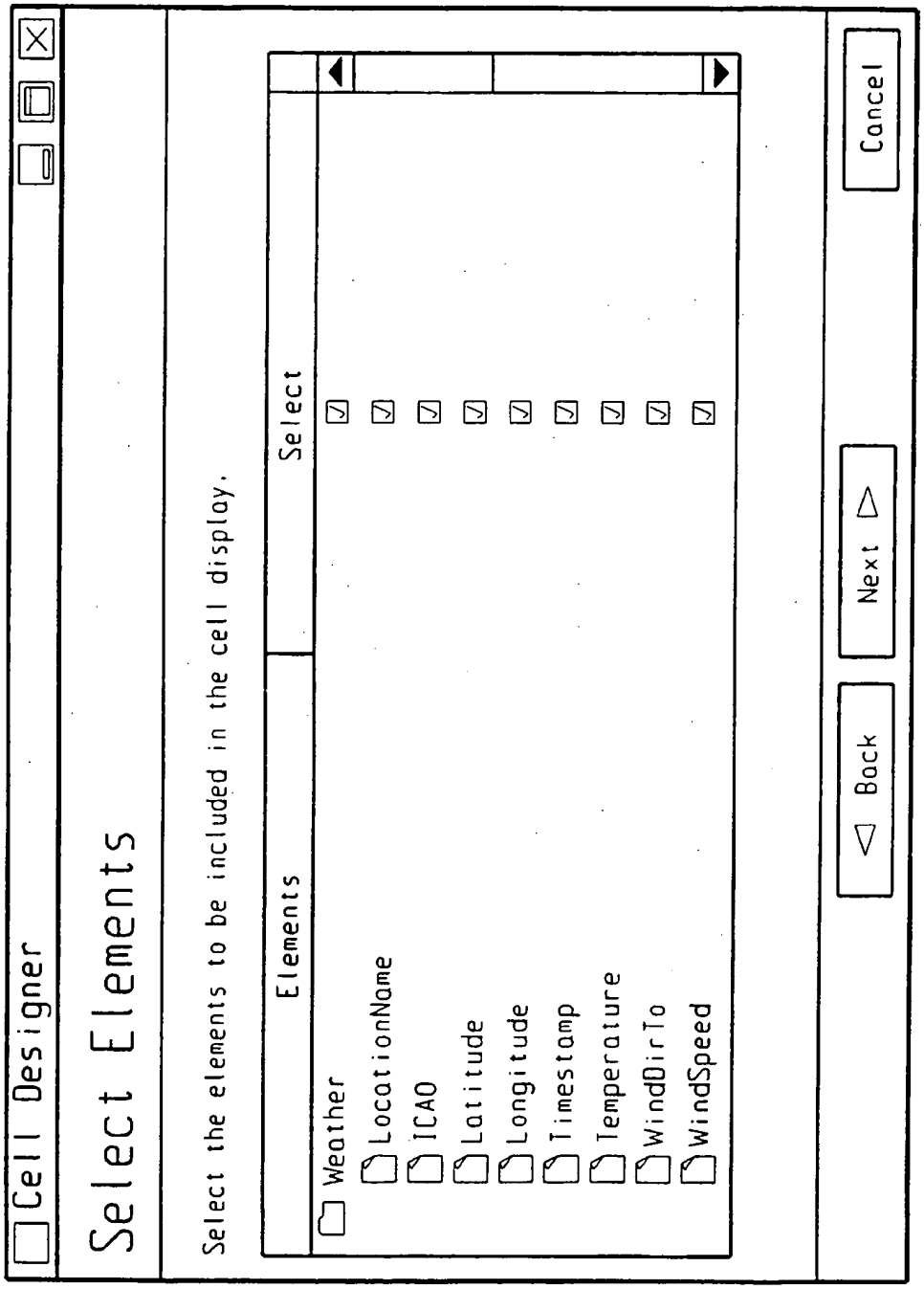

Cell Designer
Select Elements
Select the elements to be included in the cell display.
Elements
Select
Weather
LocationName
ICAO
Latitude
Longitude
Timestamp
Temperature
WindDirTo
WindSpeed
Back
Next
Cancel

Fig. 9d

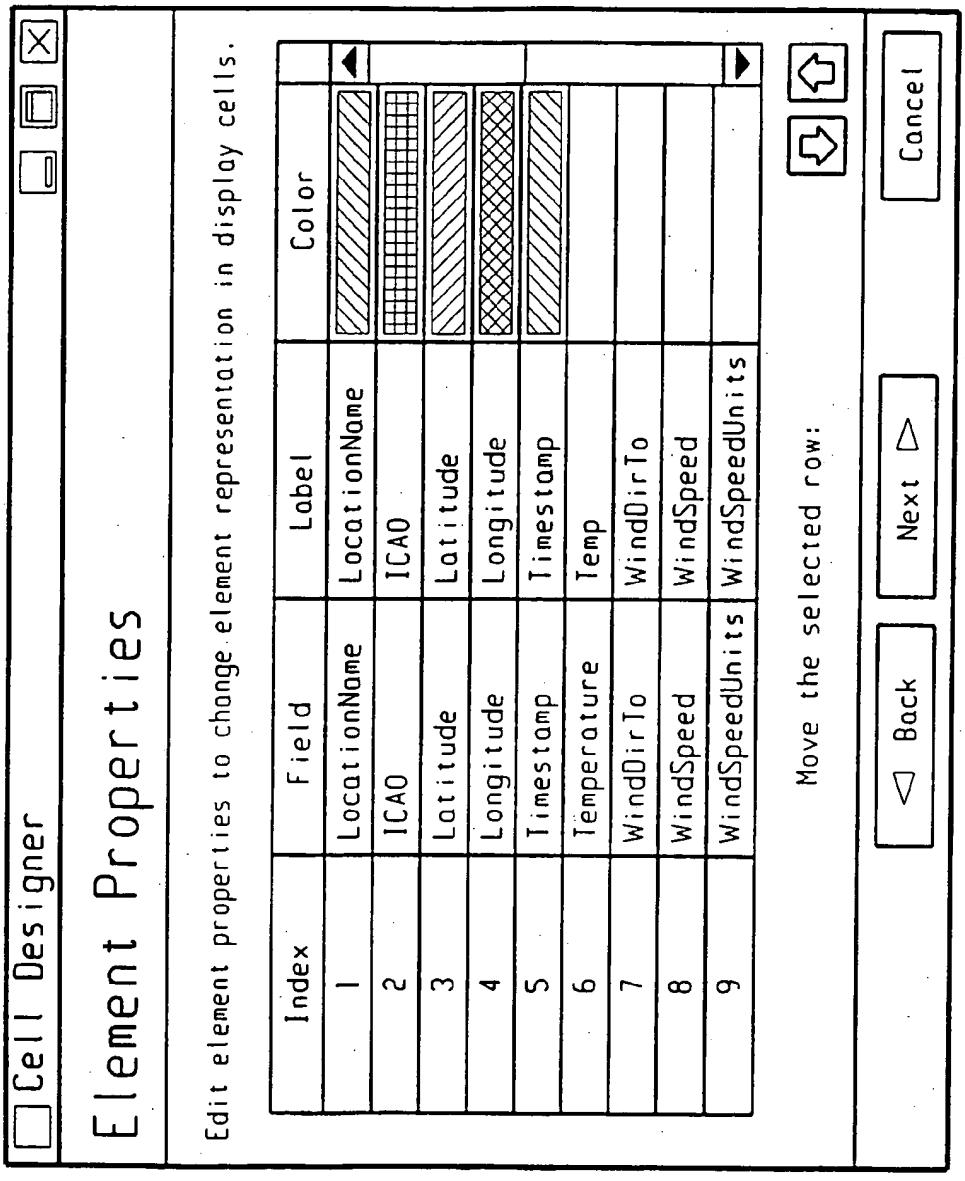

Cell Designer
Element Properties
Edit element properties to change element representation in display cells.
Index
Field
Label
Color
1
LocationName
LocationName
2
ICAO
ICAO
3
Latitude
Latitude
4
Longitude
Longitude
5
Timestamp
Timestamp
6
Temperature
Temp
7
WindDirTo
WindDirTo
8
WindSpeed
WindSpeed
9
WindSpeedUnits
WindSpeedUnits
Move the selected row:
Back
Next
Cancel

Fig. 9e

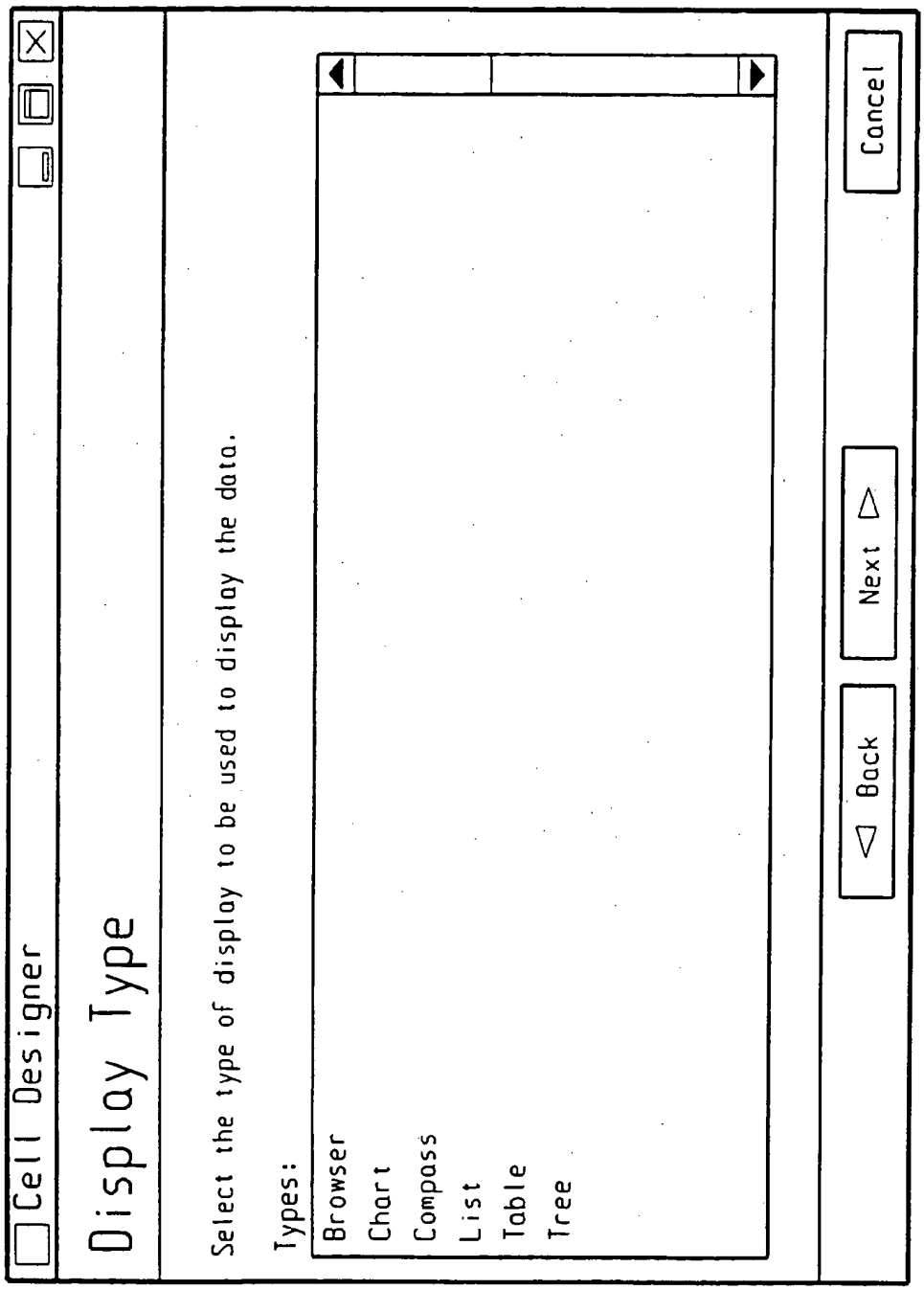

Cell Designer
Display Type
Select the type of display to be used to display the data.
Types:
Browser
Chart
Compass
List
Table
Tree
Back
Next
Cancel

Fig. 9f

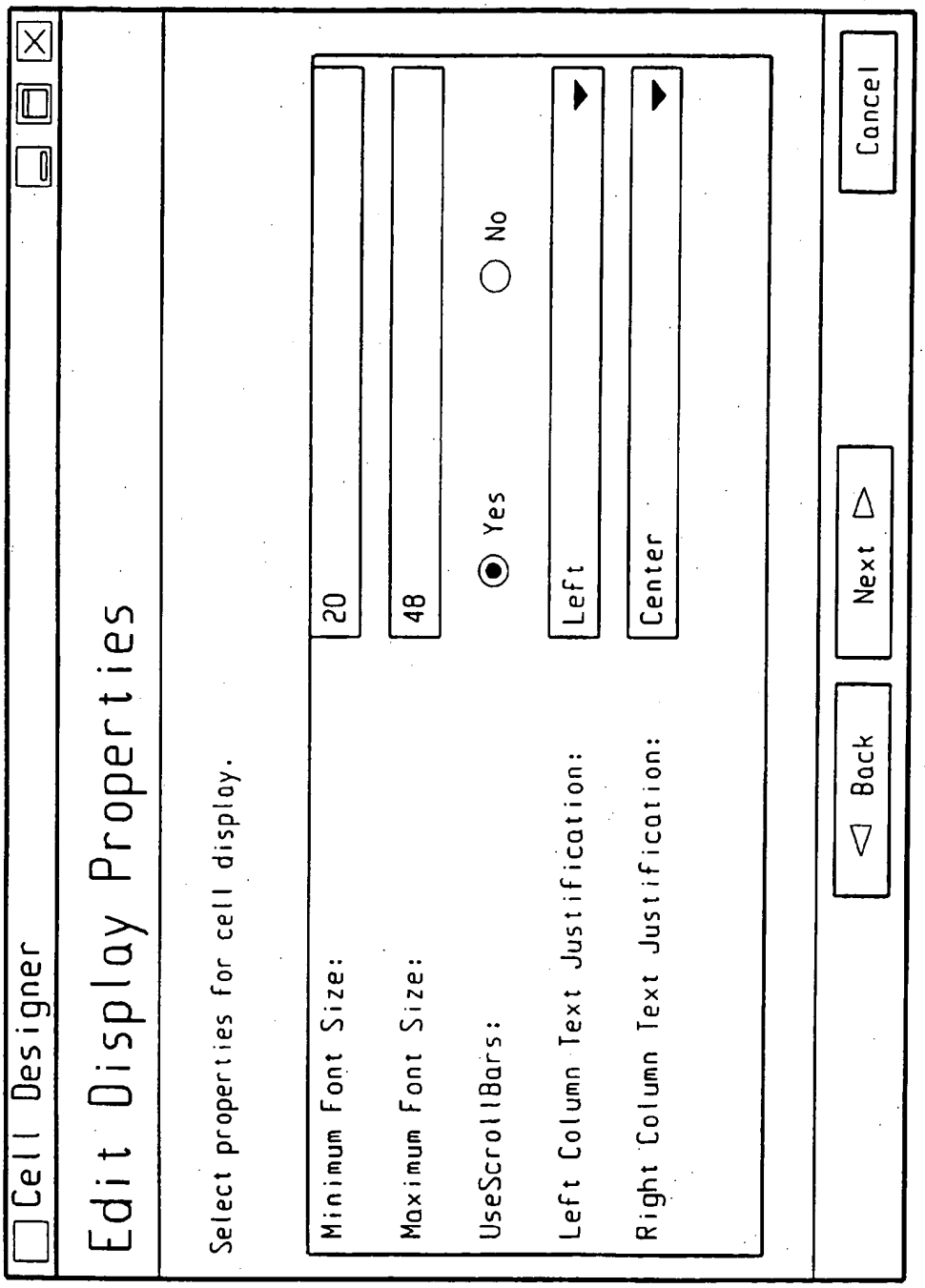

Cell Designer
Edit Display Properties
Select properties for cell display.
Minimum Font Size:
20
Maximum Font Size:
48
UseScrollBars:
Yes
No
Left Column Text Justification:
Left
Right Column Text Justification:
Center
Back
Next
Cancel

Fig. 9g

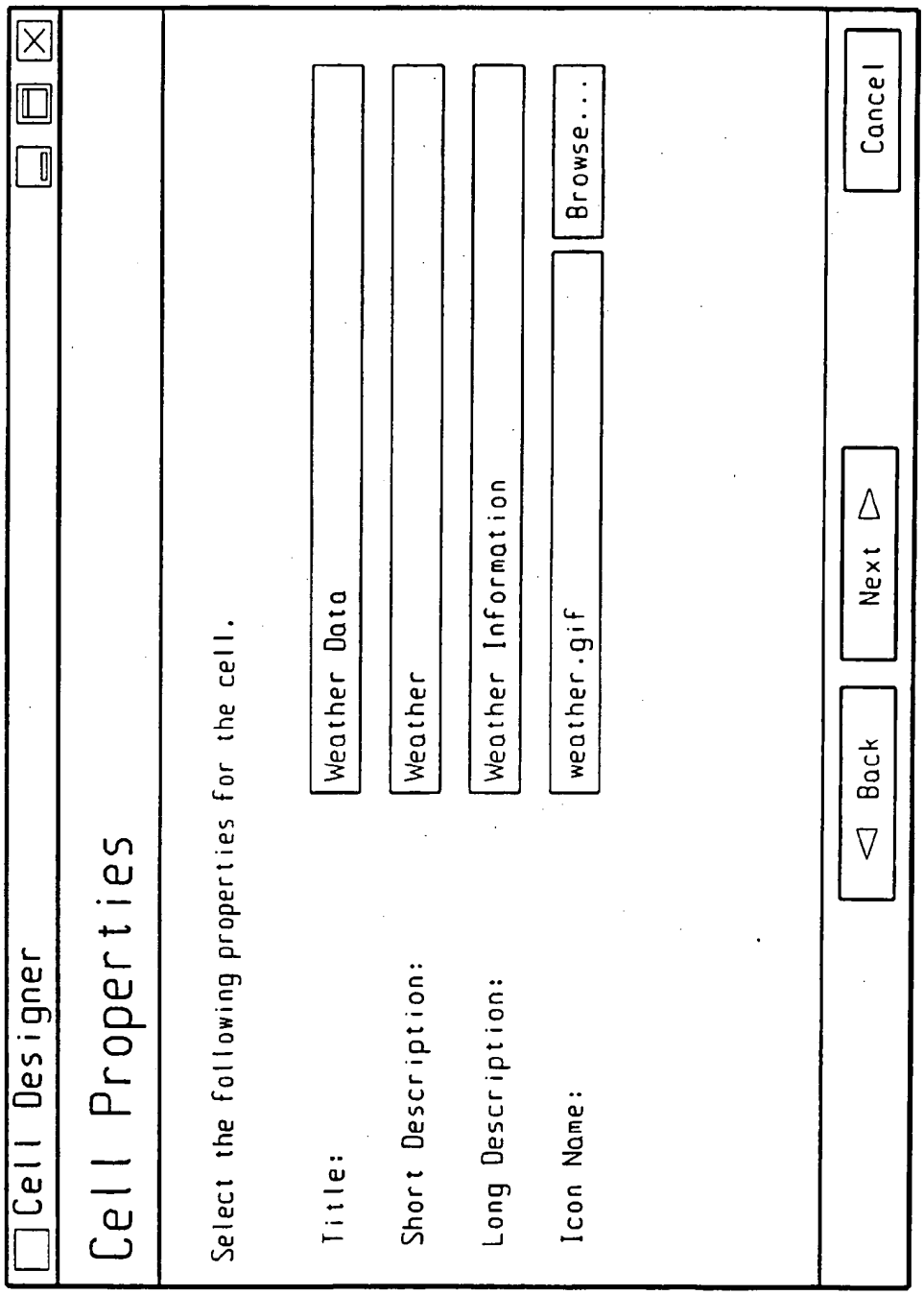

Cell Designer
Cell Properties
Select the following properties for the cell.
Title:
Weather Data
Short Description:
Weather
Long Description:
Weather Information
Icon Name:
weather.gif
Browse...
Back
Next
Cancel

Fig. 9h

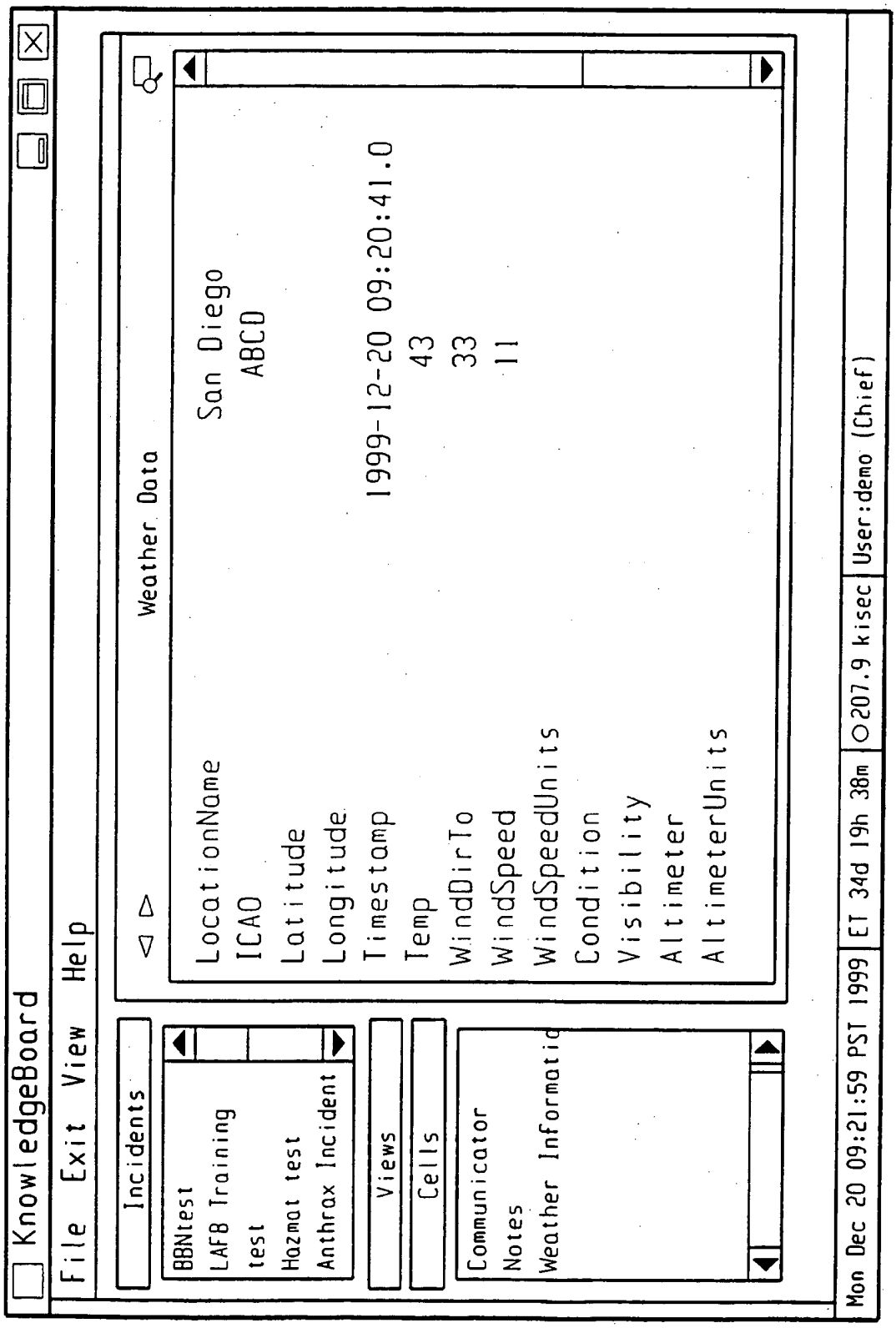

KnowledgeBoard
File Exit View Help
Incidents
BBNtest
LAFB Training
test
Hazmat test
Anthrax Incident
Views
Cells
Communicator
Notes
Weather Informatio
Weather Data
LocationName
ICAO
Latitude
Longitude
Timestamp
Temp
WindDirTo
WindSpeed
WindSpeedUnits
Condition
Visibility
Altimeter
AltimeterUnits
San Diego
ABCD
1999-12-20 09:20:41.0
43
33
11
Mon Dec 20 09:21:59 PST 1999
ET 34d 19h 38m
207.9 kisec
User:demo (Chief)

Fig. 9i